VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Měření teploty, atmosférického tlaku a hmotnosti a následný přenos dat do mobilního telefonu

Measuring of Temperature, Atmospheric Pressure and Weight and Data Transmission to Mobile Phone

Vojtěch Vávra

Bakalářská práce

Vedoucí práce: Ing. Pavel Nevlud

Ostrava, 2021

Abstrakt

Cílem této bakalářské práce je sběr dat v pravidelných časových intervalech z monitorovaného včelího úlu, přenos naměřených dat do mobilního telefonu pomocí MQTT protokolu a jejich následné vyobrazení v grafech. Teoretická část se bude zabývat popisem senzorů teploty, atmosférického tlaku a hmotnosti. Dále pak popisem komunikačních rozhraní a protokolů. V praktické části bude popsána realizace, zapojení a testování zapojených senzorů k mikropočítači a následný přenos naměřených dat na MQTT broker server a mobilního telefonu. Na konci budou vyobrazena naměřená data v grafech.

Klíčová slova

Arduino; LoRa; MQTT; měřící senzory; teplota; tlak; vlhkost; hmotnost; WeMos; WiFi

Abstract

The aim of this bachelor thesis is to collect the data in regular time intervals from monitored bee hive, transmission of collected data to mobile phone using MQTT protocol and its display in charts. The theoretical part aims to describe the senzors of temperature, atmospheric pressure and weight. There is also description of communication interfaces and protocols. In practical part there is description of realization, connection and testing of used sensors to microcomputer and transmission of collected data to MQTT broker server and mobile phone. In the end data will be shown in charts.

Keywords

Arduino; humidity; LoRa; measuring sensors; MQTT; pressure; temperature; weight; WeMos; WiFi

Poděkování

Rád bych poděkoval svému vedoucímu práce panu Ing. Pavlu Nevludovi za odborné vedení a cenné rady. Dále bych chtěl poděkovat své rodině za vyhrazení místa v obývacím pokoji pro konstrukci a pájení praktické části a za jejich velkou podporu.

Obsah

Seznam použitých zkratek a symbolů

A	–	Ampér
AD	–	Analog/Digitál
A/D	–	Analog/Digitál
ADC	–	analog-to-digital converter (analogově digitální převodník)
Ah	–	Ampérhodina
CSV	–	Comma-separated values
F	–	Síla [N]
hPa	–	Hektopascal
I	–	Elektrický proud [A]
IoT	–	Internet of Things
IP	–	Ingress Protection (Stupeň krytí)
I^2C	–	Inter-Integrated Circuit
JSON	–	JavaScript Object Notation
l	–	Délka [m]
LCD	–	Liquid Crystal Display (Displej z kapalných krystalů)
LoRa	–	Long Range
MHz	–	Megahertz
mA	–	Miliampér
mAh	–	miliAmpérhodina
MISO	–	Master In, Slave Out
MOSI	–	Master Out, Slave In
MQTT	–	Message Queuing Telemetry Transport
N	–	Newton
QoS	–	Quality of Service (Kvalita služeb)
R	–	Elektrický odpor [Ω]
S	–	Obsah [m^2]
SCK	–	Serial Clock
SCLK	–	Serial Clock

SS	–	Slave Select
U	–	Elektrické napětí [V]
USB	–	Universal Serial Bus
V	–	Volt

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Cílem této bakalářské práce je navrhnout řešení pro pravidelný sběr dat z monitorovaného včelího úlu, zaslání naměřených hodnot na MQTT broker server, následné přeposlání na mobilní telefon a jejich vyobrazení v grafech.

Téma této bakalářské práce jsem si vybral z důvodu rozmanitosti projektu, kde mě zaujalo propojení přírody s elektronikou a kde lze uplatnit manuální zručnost. Tento výrobek může usnadnit včelařům kontrolu stavu jejich úlu a včelstva.

Za tímto účelem bude použita deska Arduino Nano, která bude umístněna u včelího úlu a k ní budou připojeny měřící senzory. Dále bude použita druhá deska WeMos D1 R2, která v sobě obsahuje WiFi modul. Tato deska bude umístněna v domě a bude zasílat data na MQTT broker server.

Při chovu včel je důležité se dobře starat, jak o úly, tak o včely. Práci může zefektivnit pravidelné monitorování úlu na dálku. Tímto způsobem lze na dálku kontrolovat stav teploty úlu. Právě teplota hraje v životě včelstva nejdůležitější roli a s tím i přítomnost zdravé kladoucí matky. Optimální teplota úlu je 30 - 35 stupňů Celsia, v zimě stačí 20 stupňů Celsia. Dalším důležitým údajem k monitorování je hmotnost úlu. Hmotností zjistíme, kdy v letních období přidat medník do úlu. Medník je nástavek, který se vkládá do včelího úlu a zde si včely ukládají suroviny - nektar nebo medovici, z čeho následně vzniká med. V zimním obdobím musíme včelám vrátit zpátky zásoby ve formě cukerného roztoku. Díky monitorované váze zjistíme, zda včely mají dostatek těchto zásob, nebo potřebují doplnit.

Ve druhé kapitole této práce budou popsány senzory teploty, atmosférického tlaku a hmotnosti. Ve třetí kapitole bude popsán komunikační protokol MQTT. Dále v této kapitole budou popsány komunikační rozhraní I^2C, SPI, USB a WiFi. Ve čtvrté kapitole bude popsán návrh zařízení pro sběr dat. V páté kapitole bude popsán návrh zařízení pro přenos dat na MQTT broker server. V šesté kapitole bude popsán průběh testování, sběr dat a dále zde budou zobrazeny některé grafy. V závěrečné šesté kapitole budou shrnuty výsledky práce a v krátkosti bude uvedeno její možné rozšíření.

Kapitola 2

Popis senzorů

Na úvod této kapitoly si vymezíme několik souvisejících pojmů. Dále zde budou popsány principy senzorů teploty, tlaku a hmotnosti.

Senzor je funkční prvek, který je v přímém kontaktu s měřeným prostředím a tvoří tak vstupní blok měřícího řetězce.

Pojem senzor je ekvivalentní pojmu snímač, převodník nebo detektor. Citlivá část senzoru se v některých případech označuje jako čidlo. Senzor jako primární zdroj informace snímá sledovanou fyzikální, chemickou nebo biologickou veličinu a dle určitého definovaného principu ji transformuje na měřící veličinu - nejčastěji na veličinu elektrickou. Dále existují senzory, u nichž je neelektrická veličina přímo transformována na číslicový signál. Monografie je zaměřena převážně na elektrické senzory. (Stanislav Ď., 1996) [1]

Čidlo je snímací zařízení schopné něco regulovat. Detektor je zařízení zjišťující přítomnost nějakého (fyzikálního) jevu. Snímač je technické zařízení převádějící (světlo, zvuk ap.) na elektrický signál. Spínač je mechanické zařízení k vytvoření (zapnutí) nebo přerušení (vypnutí) elektrického obvodu. Hlídač je automatické pojistné zařízení signalizující nebezpečí poruchy. Hlásič je zařízení odhalující nebezpečí (požár, pohyb apod.). (Schafferová M., 2018) [2]

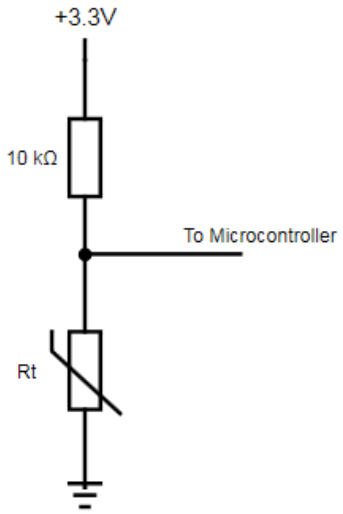
2.1 Senzor pro měření teploty

Teplota je jednou z nejdůležitějších veličin ovlivňující téměř všechny stavy a procesy v přírodě. K měření teploty se využívá celé řady funkčních principů, jejichž přehled je uveden v tabulce A.1. (Kmínek M.) [3]

Jeden ze způsobů, jak měřit teplotu, je za pomocí elektronického obvodu s termistorem zapojeným jako dělič napětí viz obr. 2.1. [4]

2.1.1 Termistor

Termistor je speciální druh odporového polovodiče, jehož odpor se silně mění s teplotou. Běžně se používá jako teplotní čidlo. Značku termistoru je možné vidět na obrázku 2.1 pod označením R_t. Měření teploty souvisí s měnícím se odporem termistoru. Arduino deska nemá vestavěný měřič odporu, proto se musí převést odpor termistoru na napětí a toto napětí číst přes analogový pin Arduina. Poté se vypočítá teplota pomocí Steinhart–Hartovy rovnice, která popisuje odporově-teplotní křivku termistoru. [4]



Obrázek 2.1: [5] Schema zapojení termistoru jako děliče napětí

2.1.2 Dělič napětí

Dělič napětí se používá tam, kde potřebujeme odebírat nižší napětí, než je svorkové napětí zdroje. Je tvořen sériovým zapojením dvou odporů nebo rezistorem s odbočkou, která jej rozděluje na dva díly s odpory R_1 a R_2, spojené do série. Dělič napětí se připojuje paralelně ke zdroji a potřebné nižší napětí U_{20} se odebírá ze středu (odbočky). Podle toho, zda odebíráme z odbočky proud, můžeme děliče dělit na:

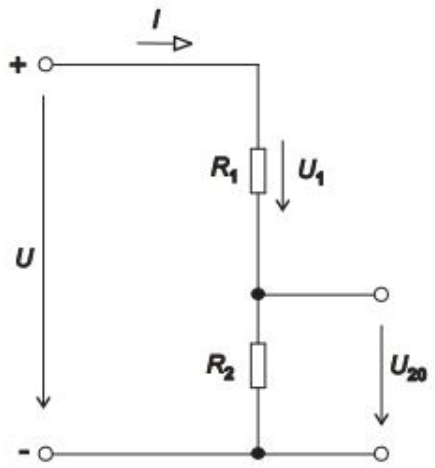
a) nezatížený dělič

b) zatížený dělič

[6] [7]

2.1.2.1 Nezatížený dělič napětí

Na obrázku 2.2 lze vidět nezatížený dělič napětí. Napětí na rezistorech se dělí v poměru jejich odporů. Z nezatíženého děliče neodebíráme žádný proud. Proud procházející oběma rezistory je stejný. Celkové napětí se rozdělí v poměru na rezistory R_1 a R_2. [6] [7]



Obrázek 2.2: [6] Nezatížený dělič napětí

Zdroj do děliče dodává proud

$$I = \frac{U}{R_1 + R_2}$$

Protože $U_{20} = R_2 \cdot I$ můžeme psát

$$I = \frac{U_{20}}{R_2}$$

Dosazením obou vztahů pro I dostaneme

$$\frac{U_{20}}{R_2} = \frac{U}{R_1 + R_2}$$

po úpravě pak

$$U_2 = U\frac{R_2}{R_1 + R_2}$$

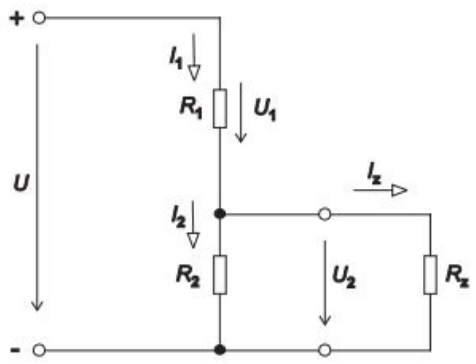
Pro poměr velikostí napětí a rezistorů platí:

$$\frac{U_1}{U_2} = \frac{R_1}{R_2}$$

[6]

2.1.2.2 Zatížený dělič napětí

Dělič je zatížen, je-li na výstup připojen spotřebič, nebo zatěžovací rezistor R_z, který odebírá proud. Schéma zapojení je znázorněno na obrázku 2.3. [7]



Obrázek 2.3: [6] Zatížený dělič napětí

Odebíraný proud musí být mnohem menší než proud procházející děličem, aby napětí na výstupních svorkách příliš nekleslo. Proud odebíraný ze zdroje děličem:

$$I_1 = \frac{U}{R_1 + \frac{R_2 R_z}{R_2 + R_z}}$$

kde ve jmenovateli je vyjádřen celkový odpor zařízení. Proud I_z dodávaný děličem do zátěže R_z je podle prvního Kirchhoffova zákona: $I_z = I_1 - I_2$.

Z Ohmova zákona je:

$$U = (R_1 + \frac{R_2 R_z}{R_2 + R_z}) I_1$$

Protože $I_1 = I_2 + I_z$ dostáváme:

$$U_2 = \frac{R_2 R_z}{R_2 + R_z} I_1$$

Vyjádříme-li z obou rovnic I_1 a položíme-li dané vztahy do rovnosti, dostáváme po úpravě pro svorkové napětí mezi výstupními svorkami při zatížení:

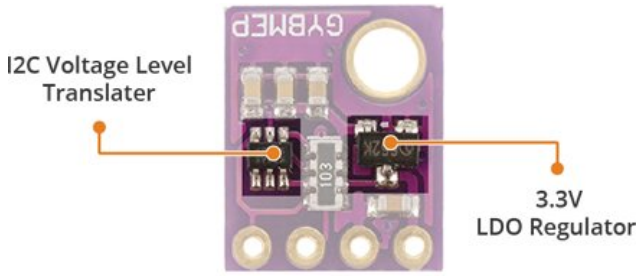
$$U_2 = U \frac{R_2 R_z}{R_1 R_2 + R_1 R_z + R_2 R_z}$$

[6]

2.1.3 Modul BME280

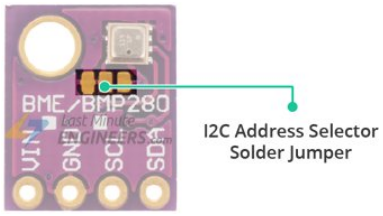
BME280 modul, který vidíte na obázku 2.4 slouží pro měření teploty, barometrického tlaku a vlhkosti. Obsahuje LM6206 regulátor napětí na 3,3 V a převodník logických úrovní. Proto lze tento

modul napájet 3,3 V, nebo 5 V napětím a použít k tomu 3,3 V nebo 5 V logiku. Díky tomu lze tento modul snadno použít s Arduino Nano, Arduino Uno, které používají 5 V logiku a není třeba připojovat externí převodník logických úrovní. [8]

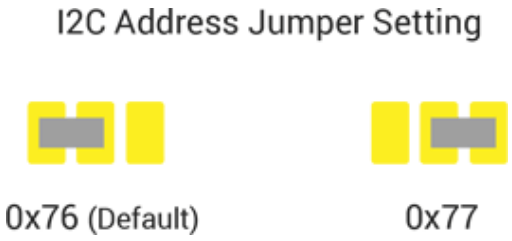


Obrázek 2.4: [8] Modul BME280

BME280 komunikuje pomocí I^2C sběrnice. Více o tomhle komunikačním rozhraní je popsáno v kapitole 3.2. Výchozí I^2C adresa tohoto modulu je 0x76. Tato adresa může být změněna na 0x77 pomocí propojů na desce viz obrázek 2.5. Možné kombinace těchto propojů pro výběr I^2C adresy jsou znázorněny na obrázku 2.6. [8]



Obrázek 2.5: [8] Modul BME280



Obrázek 2.6: [8] I2C adress jumper

2.2 Senzor pro měření atmosférického tlaku

Atmosférický tlak nám říká, jakou silou působí vzdušný obal Země (atmosféra) kolmo na její povrch.[9]

Tlak je síla, která působí kolmo na určitou plochu. Tlak označujeme písmenem p. Základní jednotkou tlaku je pascal (Pa). 1 pascal je tlak, který vyvolá síla 1 N rovnoměrně rozložená na ploše s obsahem $1m^2$, kolmé ke směru síly. Vedlejší jednotky jsou kPa (kilopascal), MPa (megapascal), hPa (hektopascal). Tlak se může měřit například barometrem (tlakoměrem). Vzorec pro výpočet tlaku:

$$p = \frac{F}{S}$$

Kde F je síla [N], která působí kolmo na rovinnou plochu o obsahu S [m^2]. S je obsah rovinné plochy. Působení síly F je rovnoměrně rozloženo na plochu o obsahu S. [10]

Při rostoucí výšce tlak klesá (např. přibližně ve výšce 5,5 km působí tlak jen asi 500 hPa), protože se zmenšuje sloupec atmosféry nad zemí. Změnu tlaku vzduchu na 100 m výšky udává tzv. vertikální tlakový gradient (na každých 100 m klesá tlak vzduchu v nižších polohách o přibližně 13 hPa - přesná hodnota se liší podle charakteristiky vzduchové hmoty). Tlak je taky ovlivňován teplotou, kde s teplejší vzduchovou hmotou tlak s výškou klesá pomaleji. [9]

Nulový tlak je prostor bez hmotnosti (vakuum). Absolutní tlak je měřen od nuly. Statický tlak zemského obvzduší měřený u zemského povrchu se nazývá atmosférický tlak. Rozdíl hodnot dvou současně působících tlaků je tlak rozdílový. [1] V elektrotechnice se dá tlak měřit mimo jiné následujícími způsoby:

- Odporové měření tlaku
- Kapacitní měření tlaku
- Piezorezistivní měření tlaku

2.2.1 Odporové měření tlaku

Princip odporového měření tlaku je založeno na měření změny v odporu elektrických vodičů způsobené tlakovou výchylkou. Následující rovnice se aplikuje pro odpor elektrického vodiče:

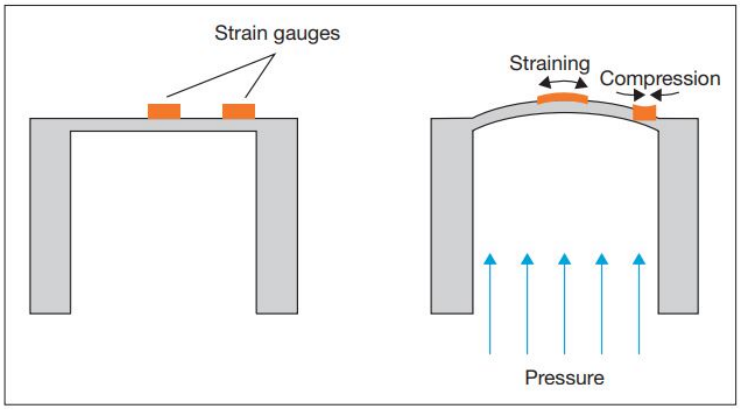
$$R = \rho \cdot \frac{l}{S}$$

Kde R je elektrický odpor, ρ rezistivita - měrný elektrický odpor vodiče, l délka vodiče, S obsah průřezu vodiče.

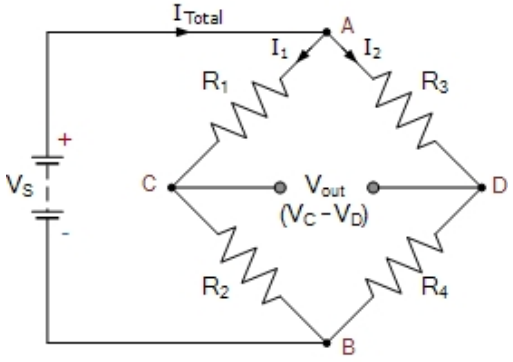
Pokud se tahová síla aplikuje na vodič, jeho délka vzrůstá a plocha průřezu klesá, tím se elektrický odpor zvětšuje. V případě komprese se délka vodiče zmenšuje, průřez vodiče zvětšuje a elektrický odpor klesá.

Princip měření tlaku pomocí elektrického odporu je realizován hlavním tělem, které vykazuje řízenou výchylku pod tlakem. Toto hlavní tělo má často tenkou oblast označovanou jako membrána, která je záměrně oslabována. Stupeň výchylky způsobený tlakem se měří pomocí metalických tenzometrů.

Na membránu se obvykle používají čtyři tenzometry. Některé z nich jsou umístněny na podlouhlých a jiné na stlačených plochách membrány. Pokud se membrána vychýlí pod působením tlaku, tak se i tenzometry vychýlí odpovídajícím způsobem. Tenhle princip je znázorněn na obrázku 2.7. Elektrický odpor se zvyšuje, nebo snižuje úměrně s průhybem (prodloužení, nebo stlačení). Pro přesnější měření změn v odporu jsou tenzometry připojeny do měřícího Wheatstoneového můstku, který je znázorněn na obrázku 2.8. [11]



Obrázek 2.7: [11] Vychýlení membrány snímače pod tlakem



Obrázek 2.8: [12] Wheatstoneův můstek

2.2.2 Kapacitní snímače tlaku

Základem kapacitního snímače je dvou nebo víceelektrodový systém, jehož parametry se mění v důsledku působení měřené neelektrické veličiny. Uvažujeme-li jednoduchý deskový kondenzátor, bude pro jeho kapacitu C platit (Kadlec K., 2005) [13]

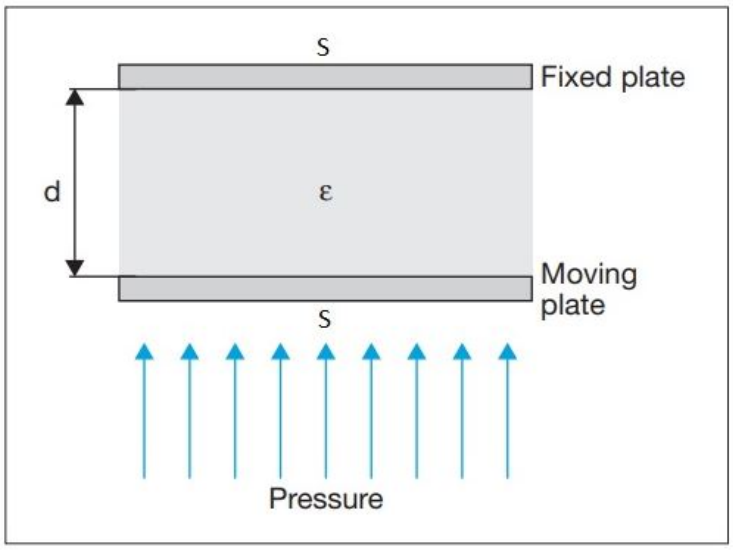
$$C = \epsilon \cdot \frac{S}{d}$$

kde ϵ je permitivita prostředí mezi deskami, S je plocha elektrod, d vzdálenost mezi deskami.

$$\epsilon = \epsilon_r \cdot \epsilon_0$$

Kde ϵ_r je relativní permitivita, ϵ_0 je permitivita vakua.

Působením neelektrických veličin se může u kapacitního snímače měnit vzdálenost mezi deskami, plocha desek nebo dielektrikum. Pro měření tlaku se využívá kapacitního snímače, u něhož dochází ke změně vzdálenosti mezi deskami. Např. jedna elektroda kondenzátoru je pevná a druhá je tvořena membránou, jež mění svou polohu v důsledku působení měřeného tlaku (Kadlec K., 2005) [13]. Princip je znázorněn na obrázku 2.9.



Obrázek 2.9: [11] Princip kapacitního měření tlaku

2.3 Senzor pro měření hmotnosti

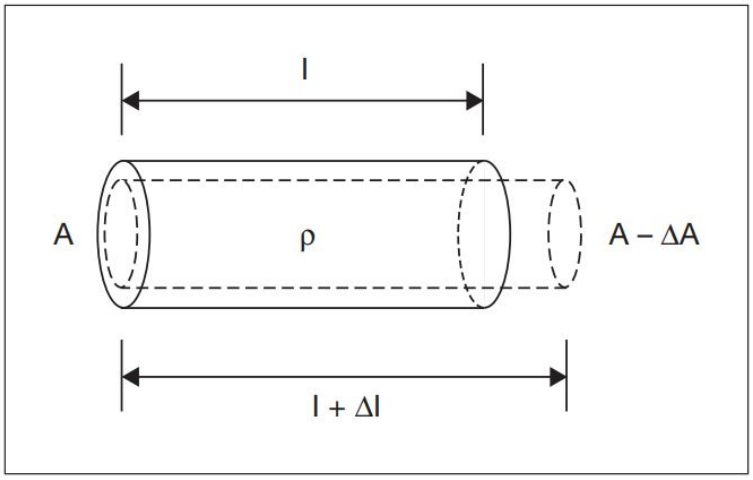
Hmotnost se považuje za základní fyzikální veličinu v soustavě SI jednotek. Jednotkou hmotnosti je kilogram (kg).

První tenzometry byly kovové drátkové. V roce 1952 byl vynalezen tenzometr fóliový. Tento tenzometr se ve velkém množství využívá až dodnes. Existují i polovodičové (křemíkové) tenzometry, které se vyznačují větší citlivostí vůči změně délky, ale zase linearitou a přesností jsou mnohem lepší fóliové tenzometry. [14]

2.3.1 Princip odporových tenzometrů

Při užití odporových tenzometrů se využívá změny odporu mechanicky namáhaného vodiče délky l_0 s průřezem S a rezistivitou ρ. Tento jev objevil již v roce 1843 pan Wheatstone. Změny odporu jsou zde nejčastěji dány změnou délky vodiče o Δl. Tenzometr je tak nejcitlivější na deformaci (natažení, prohnutí apod.) právě ve směru delší strany. Obrázek 2.10 zobrazuje změnu délky vodiče l a průřez S (v obrázku značen jako A z anglického slova Area). Následující rovnice popisuje elektrický odpor vodiče v závislosti na její délce a průřezu. [14]

$$R = \rho \cdot \frac{l}{S}$$

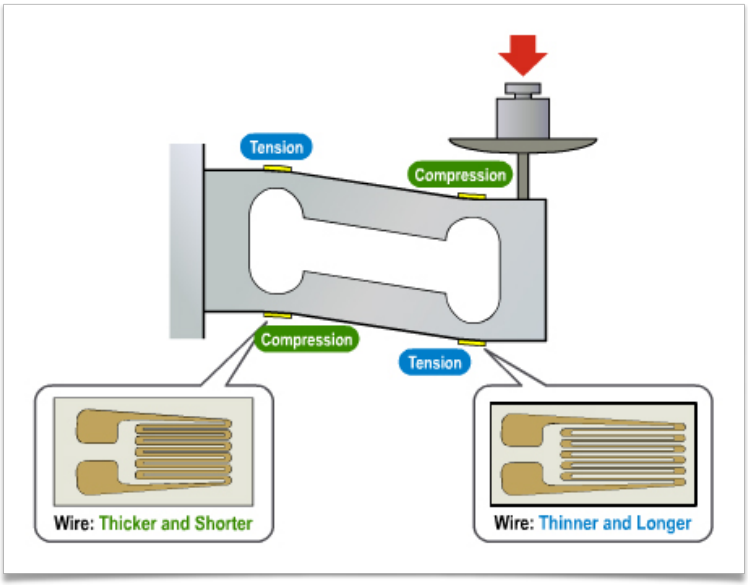


Obrázek 2.10: [11] Změna rozměrů válcového vodiče prodloužením

Na obrázku 2.11 je zachycena zátěžová tyč, která je prohnutá pod přiloženým závažím a na níž jsou přidělány čtyři tenzometrické snímače. Tyto čtyři tenzometrické snímače jsou zapojené do Wheatstoneového můstku, a díky tomu můžeme měřit malé změny odporu tenzometrů. Wheatstoneový můstek je vidět na obrázku 2.8. [15]

2.3.2 Převodník a zesilovač HX711

HX711 je analogově digitální převodník (ADC) a zesilovač vytvořený speciálně pro váhy s tenzometry, jelikož přímé propojení tenzometru s Arduino Nano analogovým pinem je pro přesné měření nedostačující. Arduino analogový pin má A/D převodník s 10-bitovým rozlišením. [16] To znamená, že Arduino mapuje 0 až 5 voltů na číselné hodnoty 0 až 1023 ($2^{10} - 1$). Mapování napětí na číselnou hodnotu je zobrazeno na obrázku 2.12. Avšak reálný průběh by nevypadal jako čistá sínusovka, který je vysvětlován na předchozích obrázcích, ale průběh by závisel na měřených hodnotách viz příklad obrázku 2.13 [17]. Podle následujícího vztahu se vypočítá, jaké bude rozlišení, velikost kroku:



Obrázek 2.11: [15] Prohnutá zátěžová tyč, která už obsahuje čtyři tenzometry zapojené do Wheatstoneového můstku

Kde stepsize je rozlišení, velikost kroku

$$stepsize = \frac{V_{ref}}{2^{rozlišení} - 1}$$

Po dosazení 10-bitového rozlišení analogového pinu Arduina:

$$stepsize = \frac{5}{2^{10} - 1} = 0,00488V = 4,88mV$$

Po dosazení 24-bitového rozlišení HX711 převodníku dostáváme stepsize, rozlišení:

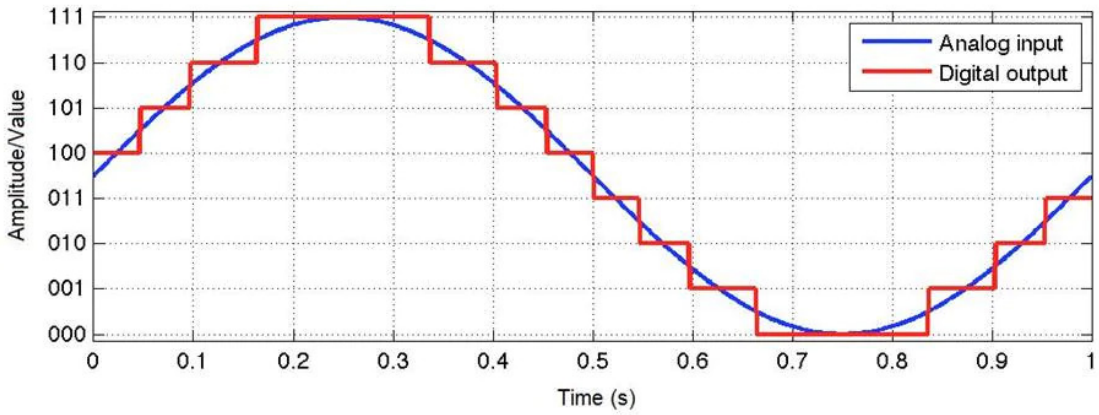
$$stepsize = \frac{5}{2^{24}} = 0,00000029V = 0,298\mu V$$

[18]

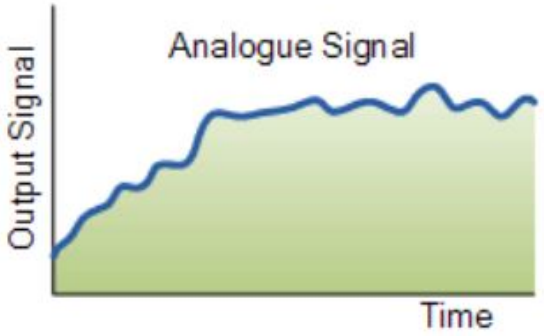
To znamená, že HX711 umí detekovat změny malé 0,298 μ, zatímco arduino analogový pin pouze 4,88 mV. HX711 mapuje 0 až 5 voltů na digitální čísla 0 až 16 777 215 ($2^{24} - 1$), zatímco Arduinu jen od 0 do 1023. [16]

K přečtení analogového vstupu na Arduinu je třeba okolo 100 mikrosekund (0.0001 s), takže maximální rychlost čtení je okolo 10000 krát za sekundu. [17] Frekvence měření HX711 je okolo 10 až 80 krát za sekundu. Napájení HX711 je 2,6 - 5,5 V a proudový odběr menší než 10 mA. [19]

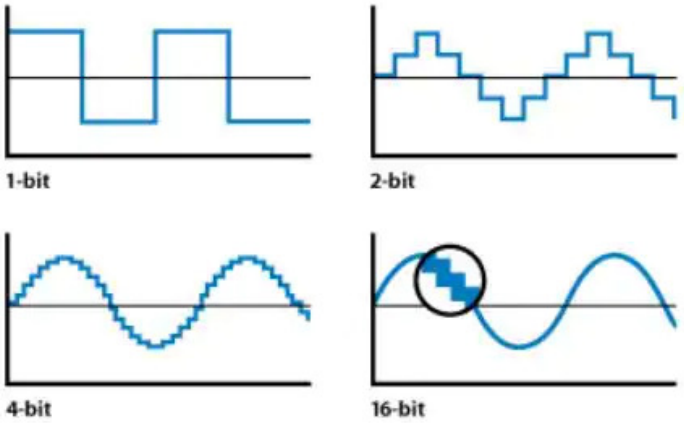
Ukázka čtení analogového signálu na digitální s různým rozlišením je znázorněna na obrázku 2.14.



Obrázek 2.12: [20] Mapování ADC signálu



Obrázek 2.13: [21] Ukázka analogového signálu



Obrázek 2.14: [20] Dopad převedeného analogového signálu na digitální s různým rozlišením

Kapitola 3

Popis komunikačních rozhraní a protokolů

V této kapitole bude popsán komunikační protokol MQTT a komunikační rozhraní I^2C, SPI, USB a WiFi.

3.1 Komunikační protokol MQTT

MQTT (dříve nazýváno: Message Queuing Telemetry Transport, dnes MQ Telemetry Transport) je protokol pro posílání zpráv ve světě Internetu věcí (IoT). Jedná se o jednoduchý a datově nenáročný protokol pro posílání zpráv a jejich přijímání, založený na architektuře publikovatel - odběratel (publisher – subscriber) [22]. Díky tomu, že je tento protokol nenáročný a jednoduchý je snadno implementovatelný i do zařízení s „malými“ procesory a poměrně rychle se rozšířil.

Přenos probíhá pomocí TCP/IP. Existuje zde jeden centrální bod (MQTT broker), který se stará o výměnu zpráv [23]. MQTT broker je server, na který chodí veškeré zprávy od publikovatelů (publisherů) a dále je pak rozesílá odběratelům. Všechny zprávy tedy musejí projít přes brokera před tím, než mohou být doručeny. [24]

Zprávy se třídí do tzv. témat (topics) a zařízení buď publikuje v daném tématu (publish), to znamená, že posílá data brokeru, který si je ukládá a dále distribuuje dalším zařízením, nebo je zařízení, nebo uživatel přihlášen k odběru (subscribe) tématu, nebo i více témat, a broker pak všem klientům, nebo zařízením zasílá zprávy, které odebírají. Jedno zařízení může najednou být jako publisher i odebíratel a to i v různých tématech

Obsah zasílaných zpráv není nijak definovaný ani vyžadovaný. Jsou to prostá binární data, které se zasílají. Velikost zprávy je protokolem omezena na necelých 256 MB, ale naprostá většina zpráv bývá mnohem menší. [23]

3.1.1 MQTT Komunikace na jednotlivých QoS úrovních

MQTT protokol dává klientovi na výběr mezi třemi úrovněmi kvality služeb - QoS (Quality of Service). Tyto tři úrovně určují, jak moc se má dbát na spolehlivost doručení zpráv. I když je vyšší úroveň QoS spolehlivější, mívají vyšší latence a větší požadavky na šířku pásma.

- QoS0 - nejjednodušší úroveň. Nepoužívá se zde žádného mechanismu s potvrzením. Když klient (publisher) odešle PUBLISH paket brokeru, broker rozešle zprávu všem odběratelům. Při této komunikaci se zasílá vše pouze jednou, bez jakéhokoliv ověřování, zda data dorazila v pořádku, či nikoli.

- Druhá úroveň QoS1 využívá potvrzujícího paketu. Využívá se sekvencí PUBLISH/PUBACK. Když klient (publisher) odešle PUBLISH paket brokeru, tak následně klient vyčká, než mu broker odpoví s PUBACK. Broker po přijetí PUBLISH od klienta (publishera) odešle všem odběratelům PUBLISH zprávu, jakmile broker od odběratelů na zpět obdrží PUBACK, tak původnímu publisherovi zasílá PUBACK. Potvrzující paket PUBACK ověřuje, zda obsah byl obdržen. Pokud potvrzující paket nebyl v časovém intervalu obdržen, celá zpráva se odesílá znovu. Toto může mít za následek, že odběratel obdrží více kopií stejné zprávy. [24]

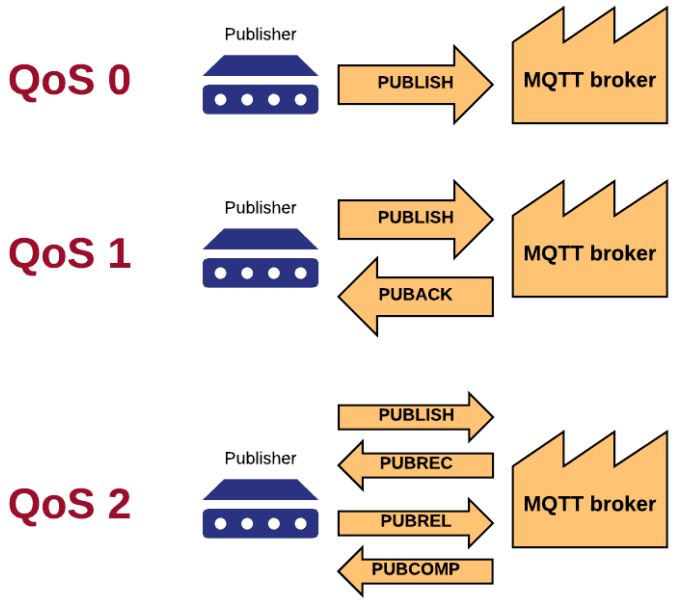
- Třetí úroveň QoS2 (dodání zprávy přesně jednou). Když broker obdrží PUBLISH zprávu od publishera, tak jí přijme a posílá ji odběratelům. Publisherovi vrátí zprávu PUBREC (potvrzení přijetí). Publisher poté odpoví zprávou PUBREL, broker následně potvrdí zprávou PUBCOMP. Tímto je výměna uzavřena. [23]

Tabulka 3.1 znázorňuje další řídící zprávy a jejich význam. Obrázek 3.1 znázorňuje komunikaci na určitých QoS úrovních.

3.2 Komunikační rozhraní I^2C

I^2C (anglicky čteno I-squared-C), někdy také nesprávně označováno jako I2C. Jde o zkratku z původního označení IIC (Inter-Integrated Circuit) Bus. Tato interní datová sběrnice slouží pro komunikaci a přenos dat mezi jednotlivými integrovanými obvody, většinou v rámci jednoho zařízení. Tato komunikační sběrnice je podporována řadou integrovaných obvodů nejen od navrhovatele této sběrnice firmy Philips. Využití této sběrnice je s propojením s mikrokontroléry, sériovými paměti, inteligentní LCD, audio a video obvody, A/D a D/A převodníky a další digitálně řízené obvody.

Největší výhodou je, že obousměrný přenos dat probíhá pouze po dvou vodičích. Jedná se o datový signál SDA (Serial Data) a hodinový signál SCL (Serial Clock). Díky tomu můžou vznikat zapojení s menším počtem pinů a celkově to zjednodušuje výsledné zapojení na desce. [25]



Obrázek 3.1: [23] MQTT QoS diagram komunikace

3.2.1 Princip komunikace I^2C rozhraní

V I^2C komunikaci bývá většinou jeden „master“ obvod (může jich být klidně i víc) a k němu připojené periferní obvody. Schéma I^2C zapojení lze vidět na obrázku 3.2. Každé připojené zařízení má, na rozdíl od SPI, vlastní sedmibitovou adresu (rozsah 0 až 127). Adresy nejsou libovolné, většinou jsou pevně dané výrobcem – či alespoň jejich část. Některé adresy jsou vyhrazené pro speciální použití I^2C rozhraní, a tak se pomalu prosazuje novější standard s desetibitovými adresami. [22]

Komunikaci zahájí master s tím, že na vývod SDA pošle logickou 0 (říkáme, že vývod „stáhne k zemi“), a poté stáhne i hodinovou linku SCL. Tím se vytvoří takzvaný „startovní signál“ (start condition). Následně master začne vysílat hodinové pulsy na SCL, a zároveň s tím nastavuje na vodič SDA data (postupně od nejvyššího bitu k nejnižšímu – zde je rozdíl proti SPI). Přenos dat vždy začíná tím, že je přenesena sedmibitová adresa (A6 – A0) a bit R/W, který říká, zda bude následovat čtení ze zařízení (1), nebo zápis do zařízení (0). Během této doby všechna zařízení naslouchají a čtou adresu vysílanou masterem. Pokud je adresa jiná, než naslouchající zařízení, tak se odpojí a čekají na STOP signál (stop condition), kdy se při SCL=1 změní SDA z logické 0 na logickou 1. Pokud je adresa stejná, zařízení se připraví k přenosu dat.

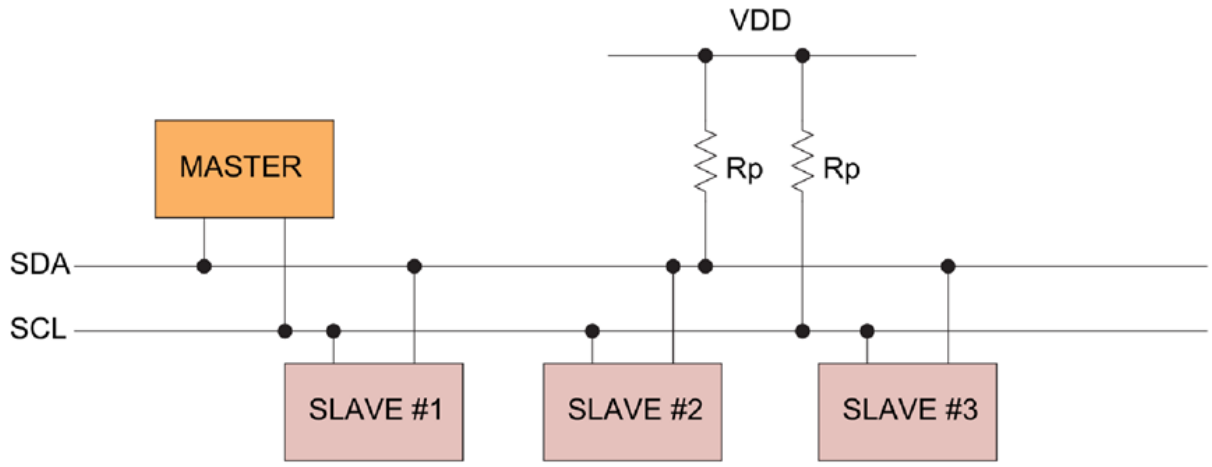
Následně master nechá datovou sběrnici odpojenou, začne naslouchat a pošle ještě jeden hodinový puls. Zařízení, které rozpoznalo svou adresu, stáhne SDA k zemi vysláním logické 0, a tím potvrdí připravenost komunikovat (říkáme tomu ACK – z anglického slova Acknowledge, neboli potvrzení). Master tento bit sleduje, a pokud přijme logickou 0, ví, že je vše v pořádku. Pokud ale

Tabulka 3.1: [24] Řídící zprávy MQTT protokolu

MQTT Zpráva	Význam
CONNECT	Požadavek klienta na připojení se k serveru
CONNACK	Potvrzení připojení
PUBLISH	Publikovat zprávu
PUBACK	Potvrzení publikování
PUBREC	Publikování přijato
PUBREL	Vydání
PUBCOMP	Publikování dokončeno
SUBSCRIBE	Žádost přihlášení klienta k odběru
SUBACK	Potvrzení přihlášení k odběru
UNSUBSCRIBE	Požadavek na odhlášení se z odběru
UNSUBACK	Potvrzení odhlášení se od tématu
PINGREQ	Ping požadavek
PINGRESP	Ping odpověď
DISCONNECT	Klient se odpojuje

během devátého hodinového pulsu najde na SDA hodnotu logickou 1, znamená to, že žádné zařízení nerozpoznalo svou adresu.

Poté následuje vlastní přenos dat, a to buď z masteru do zařízení (periferie), nebo opačným směrem. V obou dvou případech časování řídí master. Když je vše přeneseno, uvolní master sběrnici tím, že nejprve „pustí“ SCL (to znamená, že master se odpojí od SCL a pullup rezistor SCL vodič vytáhne k logické 1), a poté uvolní i SDA. Tímto způsobem se doví všechna připojená zařízení ke sběrnici, že nastal konec přenosu. [22]



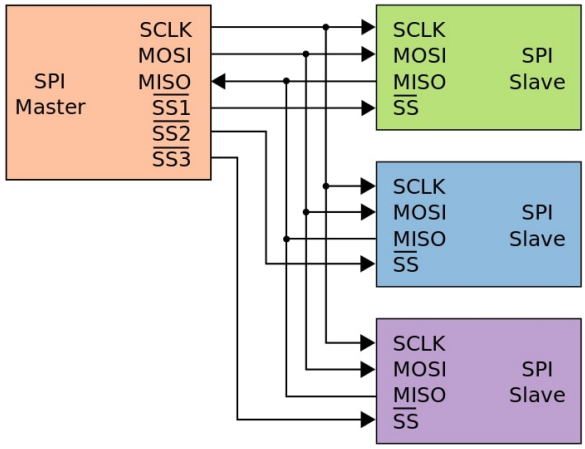
Obrázek 3.2: [26] Schéma I^2C zapojení

3.3 Komunikační rozhraní SPI

SPI z anglického Serial Peripheral Interface je sériové rozhraní, kterým bývají vybaveny různé paměti, A/D převodníky, nebo některé senzory. [22]

K tomuto rozhraní mohou být zapojeny dva a více obvodů. Připojený je obvykle procesor, který je typu Master, ostatní zařízení jsou typu Slave. Jednotlivé obvody jsou propojeny čtyřmi vodiči:

- Datový výstup MOSI (Master Out, Slave In), jenž je master propojen se všemi vstupy MOSI Slave obvodů.
- Datový vstup MISO (Master In, Slave Out), jenž je master propojen se všemi výstupy MISO Slave obvodů.
- Výstup hodinového signálu SCK (nebo označovaného taky jako SCLK) bývá propojen se všemi SCK vstupy obvodů Slave.
- Obvod Slave má vstup SS (Slave Select) propojený samostatným vodičem s každým zařízením. Pomocí tohoto vodiče si slave vybírá, se kterým zařízením chce komunikovat. Je-li SS v neaktivní úrovni, je rozhraní SPI daného obvodu neaktivní a jeho výstup MISO je ve vysokoimpedančním stavu. [27]



Obrázek 3.3: [28] Celková koncepce systému se sběrnicí SPI

3.3.1 Princip komunikace SPI rozhraní

Komunikace vypadá tak, že master vyšle signál na SS (SlaveSelect) vstup takového zařízení, se kterým chce začít komunikovat a uvede jej do logické 0.

Po výběru zařízení začne master na výstupu SCLK (Serial Clock) posílat hodinové pulsy, a zároveň na výstupu MOSI posílá data od nejnižšího bitu.

Master i slave zařízení používají posuvné registry. Master zašle do zařízení požadované informace, nejčastěji příkaz a následně parametry. Jakmile skončí přenos, uvede vstup SlaveSelect (SS) zpět do neaktivního stavu. Sběrnice SPI je plně duplexní. Díky tomu můžou v ten samý moment chodit data jak ve směru MASTER -> SLAVE, tak i v opačném, tedy z periferie do řídicího zařízení. K tomu se používá druhý vodič, MISO. [22]

3.4 Komunikační rozhraní USB

USB (Universal Serial Bus) je univerzální sériová sběrnice. Jedná se o velice častý způsob připojení periférií k počítači jako takzvaně Plug & Play, bez nutnosti restartování počítače nebo ručního instalování ovladačů. [29]

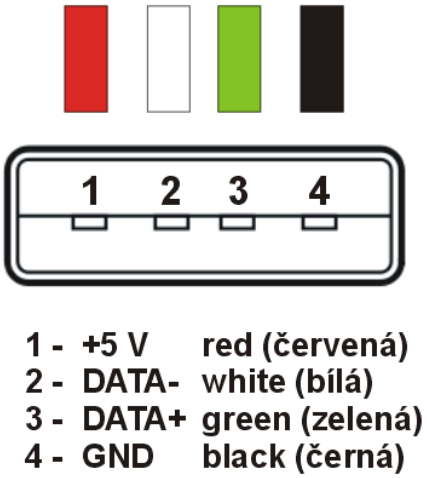
USB byla navržena s ohledem na co nejsnazší práci. Díky snadnému použití a vlastnostem postupně vytlačila prakticky všechny starší technologie, které sloužily k připojení externích zařízení k počítačům, jako třeba port pro klávesnici a myš (původně se na IBM PC jednalo o konektor typu DIN, později PS/2), paralelní porty atd. Na obrázku 3.4 lze vidět schéma zapojení konektorů USB 2.0 typ A.

Na fyzické úrovni bývají jednotlivá komunikující zařízení propojena systémem point to point (jedním kabelem jsou propojena vždy jen dvě zařízení). Oproti tomu sběrnice I^2C používá společný vodič pro přenos dat a synchronizačního hodinového signálu. Některé parametry USB:

- Komunikační rychlost od 1,5 Mbit/s do 480 Mbit/s
- Komunikační vzdálenost do 5 m
- Rozhraní obsahuje 5 V napájení
- Lze připojit až 127 zařízení pomocí jednoho typu konektoru
- USB zajišťuje správné přidělení prostředků (IRQ, DMA, ...).

[30]

Podle specifikací USB bývají všechna USB zařízení řízena jedním řadičem (Host Controller). Tento hardwarový čip využívá 7 bitovou adresu a dokáže tak spravovat až 128 adres. Řadič má napevno nastavenou adresu 0. Pro inicializaci dalších zařízení, zbývá možnost fyzicky připojit 127 zařízení. Řadič rozlišuje mezi zařízeními a rozbočovači, kde rozbočovače jsou něco jako rozdvojky, ke kterým lze připojit jiné rozbočovače nebo koncová zařízení. Ke každému koncovému zařízení je přiřazena jedna adresa. Jednu adresu má i samotný rozbočovač. [31]



Obrázek 3.4: [29] Schéma zapojení USB konektorů

3.4.1 Přenos dat

Ke komunikace mezi počítačem a zařízením jsou k dispozici tři typy paketů. Každá výměna dat začíná tím, že počítač vyšle token packet, který obsahuje popis typu a směru výměny dat, adresu USB zařízení a číslo koncové jednotky (endpoint number). Následně zařízení, které má vysílat data, vyšle datový paket nebo indikuje, že žádná data nejsou k dispozici. Na konci přenosu vysílající strana vyšle handshake packet, kterým informuje, zda přenos proběhl úspěšně. Existují zde dva typy přenosového modelu:

- Tok dat (stream) - nemá přesně definovanou strukturu.
- Zpráva (message) - využívající asynchronní přenos. Má přesnou strukturu:
  - Řídicí zpráva určená pro konfigurování poprvé aktivovaného zařízení
  - Zpráva obsahující větší objem dat (např. pro tiskárnu nebo plotter), jež je většinou segmentována do více částí
  - Zpráva s přerušením (obvykle několik bajtů), kterou spontánně vysílá zařízení, aby předalo zprávu o svém stavu (např. změna polohy myši)

[30]

3.5 Komunikační rozhraní WiFi

V obecném smyslu Wi-Fi (nebo také WiFi) označuje technologie bezdrátové sítě LAN (WLAN - Wireless Local Area Network), které pro komunikaci využívají standardy IEEE 802.11. Wi-Fi

produkty používají k přenosu dat z klientského zařízení do přístupového bodu AP (Access Point) rádiové vlny. Tato technologie původně používala frekvenci 2,4 GHz, ale od té doby se rozšířila na frekvenční pásma 5 GHz, 60 GHz a brzy i 6 GHz [32]. V České Republice je frekvenční pásmo 2,4 GHz bezlicenční pásmo [33]. Výhody WiFi:

- Mobilita – uživatel není vázán na zásuvku a kabel
- Rychlost a jednoduchost uvedení do provozu
- Nižší celkové náklady na vybudování sítě (není nutné budovat nákladné rozvody strukturované kabeláže)
- Rozšiřitelnost – vhodnou volbou a polarizací antén lze snadno a rychle nejenom zvyšovat kapacitu přístupových bodů, ale také pokrýt území podle potřeby

[34]

Kapitola 4

Návrh zařízení pro sběr dat

V této kapitole bude popsán návrh a praktické zapojení Arduina Nano a napojených senzorů. Dále bude popsáno použití LoRa RFM95W modulu s Arduinem Nano. Veškeré zdrojové kódy vycházejí z knihoven pro práci s daným modulem. V textu je vždy uveden odkaz, z jaké knihovny čerpám. Výsledné uspořádání funkčnosti a logika programu je tvořena mnou.

4.1 Blokové schéma zapojení

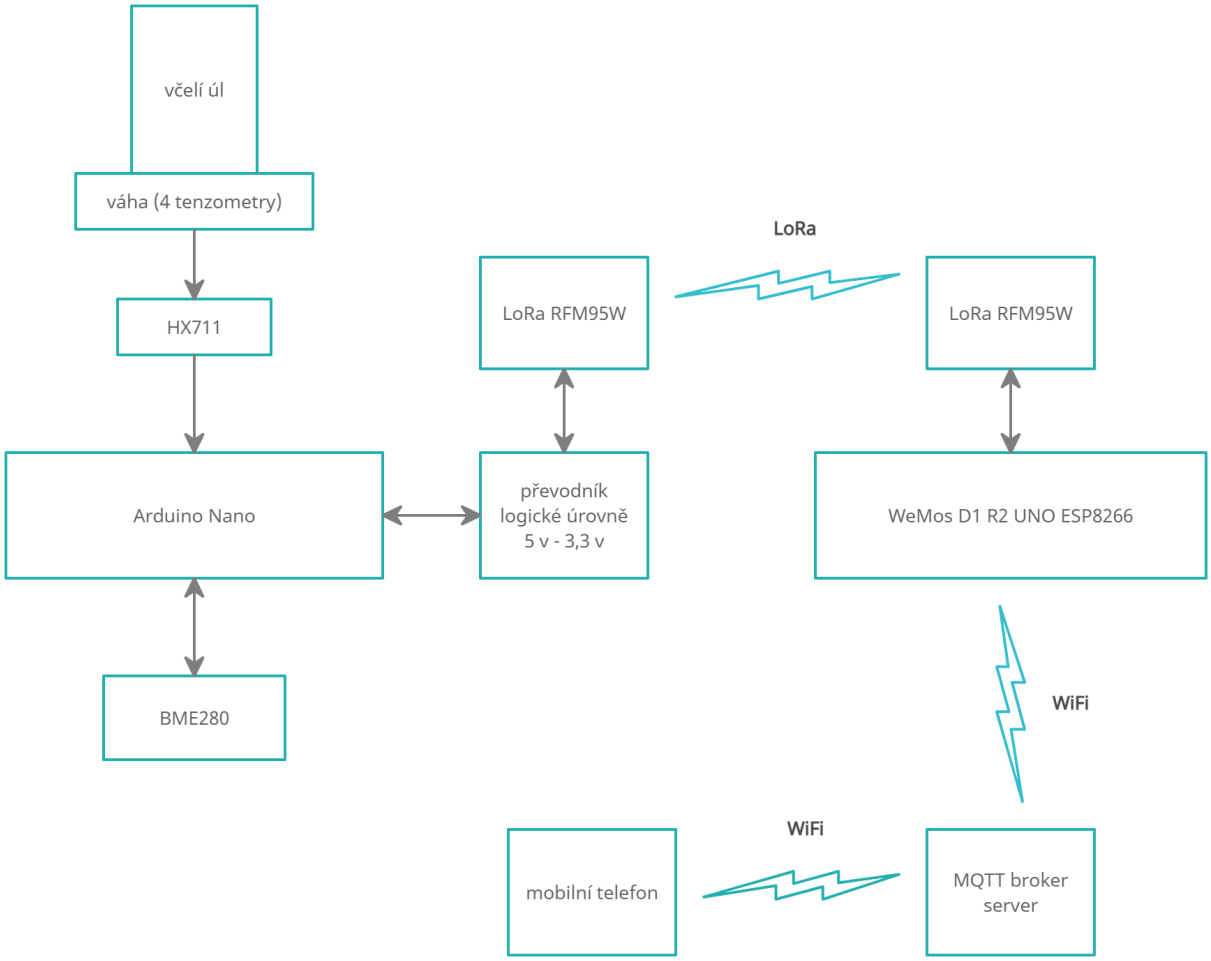
Zapojení se skládá ze dvou na sobě závislých a jedné nezávislé části. První část se stará o sběr dat v pravidelných intervalech a jejich následné zaslání přes LoRa RFM95W modul do druhé části zapojení. Druhá část přijímá přes LoRu zprávy, ty rozparsuje a přes WiFi je odesílá na MQTT broker server. Poslední nezávislou částí je přihlášení se k odběru témat na MQTT broker server pomocí mobilní aplikace a nechávat si tak do ní zasílat data. Celkové blokové schéma můžete vidět na obrázku 4.1.

4.2 Napájení Arduina Nano a připojených senzorů

Arduino Nano lze napájet buď z USB konektoru 5 V (musí být dodrženo konstantní napětí 5 V), nebo připojením 7-12 voltů napětí na VIN pin Arduina.

Při napájení Arduina Nano pomocí USB konektoru a powerbanky s výstupem 5 V a kapacitou 10 000 mAh, by přibližně v ideálním stavu powerbanka vydržela 23 dnů podle hodnot a výpočtů níže. Hodnoty odběru proudu v klidu a při zátěži byly měřeny pomocí multimetru. Postup měření je popsán níže. Senzory hmotnosti jsou napájeny z Arduina Nano pinu 3v3. Moduly BME280 a LoRa RFM95W jsou napájeny z Arduina Nano pinu +5v.

- odběr v klidu je 18 mA
- odběr při měření a odesílání dat přes LoRu po dobu zhruba 0,7 sekund je 50 mA



Obrázek 4.1: Celkové blokové schéma

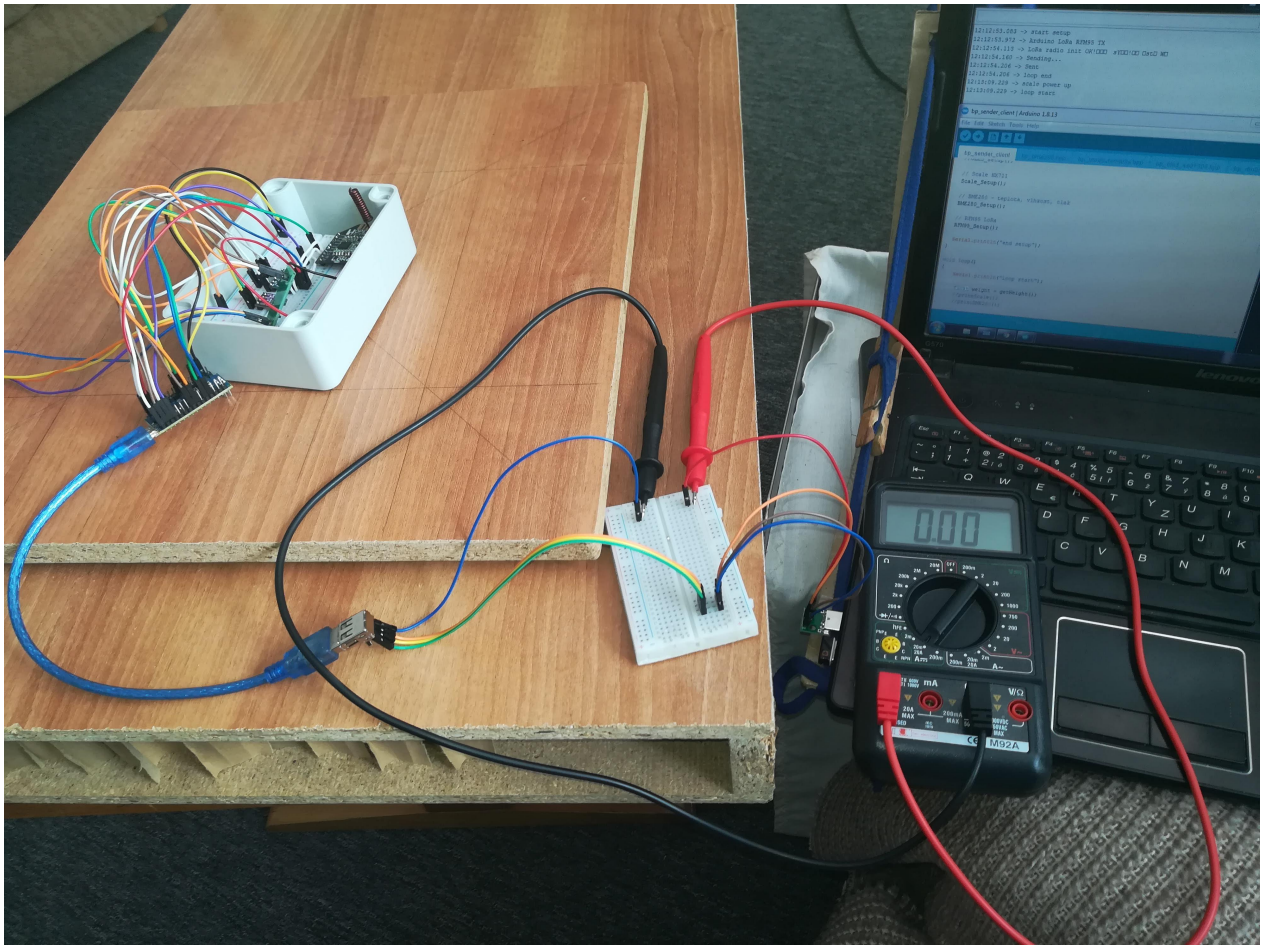
- powerbanka s kapacitou 10 000 mAh

Odběr v klidovém stavu:
10000/18 = 555.55 hodin = 23 dnů

Odběr při měření a vysílání dat přes LoRu, po dobu zhruba 0,7s:
0,7 s * 555 měření = 388 s = 0,1077 hodin
0,05 A * 0,1077 h = 0.005385 Ah = 5,385 mAh

Výsledná doba výdrže powerbanky v ideálním stavu by byla:
(10000 - 5,385) / 20 = 555.25 hodin = 23 dnů
Počet dnů výdrže powerbanky je spočítán pro ideální stav a proto při reálném nasazení by bylo potřeba vyměnit powerbanku o řádově několik jednotek dnů dříve.

Pro možnost změřit odběr v klidovém a zatíženém stavu jsem zakoupil USB Type A Plug (Male) Breakout Board a USB Type A Female konektory. Propojil jsem USB Male k USB zdířce notebooku a USB Female k USB kabelu od Arduina Nano. Dále pak přes nepájivé kontaktní pole jsem propojil USB Male s USB Female GND s GND, Data- s Data-, Data+ s Data+. Male USB pin Vcc jsem propojil s kabelem multimetru vedoucím do zdířky 20A MAX (později zdířka 200 mA MAX). Multimetr jsem nastavil na měření proudu. USB Female pin Vcc jsem propojil s kabelem multimetru do zdířky COM. Zapojení lze vidět na obrázku 4.2.



Obrázek 4.2: Měření odběru proudu Arduina Nano

4.3 Vytvoření váhy

Pro konstrukci váhy jsem použil čtyři váhové senzory (tenzometry), každý z nich vydrží maximální zatížení 50 kilogramů. Tyto senzory jsem propojil s AD převodníkem HX711. Pozici hmotnostních senzorů na desce jsem odměřil tak, aby byly všechny stejně daleko od krajů a byly díky tomu rovnoměrně zatížené. S tímto rovnoměrným rozložením lze vážit až do 200 kg, při umístění zátěže

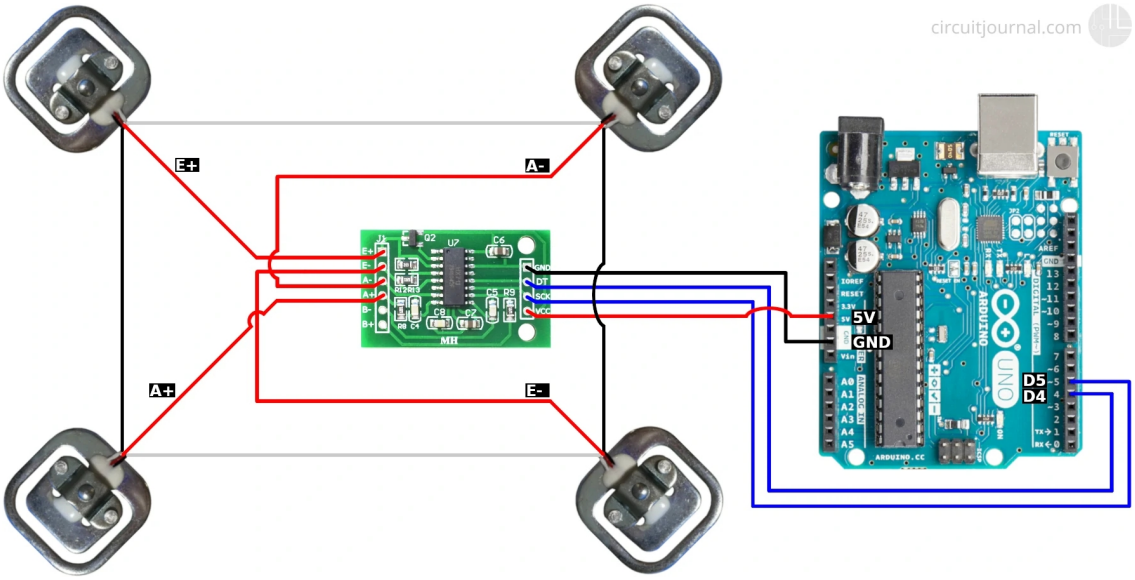
doprostřed váhové desky. Podstavu váhy o velikosti 40x40 cm jsem vyřezal z police od skříně, materiál je dřevotříska. Na spodek desky jsem přišrouboval plastová pouzdra, která drží váhové senzory, aby je bylo možné přichytit. K tomu jsem vytiskl na 3D tiskárně tento model: `https://www.thingiverse.com/thing:2624188`. Vytištěný model pouzdra je možné vidět na obrázku 4.3. Spodní část váhy je možné vidět na obrázku 4.4. Propojení pinů převodníku HX711 s Arduinem je popsáno v tabulce 4.1. Schéma zapojení Arduina s převodníkem a tenzometrickými snímači je k vidění na obrázku 4.5.

Po zapojení bylo nutné váhu zkalibrovat. Pro komunikaci s HX711 převodníkem jsem použil knihovnu `https://github.com/RobTillaart/HX711`. Jako výsledný kalibrační faktor jsem zvolil 2280, který dále používám v kódu pro měření váhy D.3.

Tabulka 4.1: Zapojení pinů převodníku HX711 s Arduinem Nano

HX711	Arduino
VCC	+5v
SCK	D5
DT	D4
GND	GND

Obrázek 4.3: Plastové pouzdro se šroubky



Obrázek 4.5: [35] Schéma zapojení váhových senzorů s HX711 převodníkem a Arduinem Uno

4.4 Modul pro měření teploty, vlhkosti a tlaku

Pro měření teploty, vlhkosti a tlaku jsem použil modul BME280. Napájím ho z Arduina 5 V pinu a komunikuji s ním přes I^2C rozhraní, které je popsáno v kapitole 3.2. V mém případě bylo nutné si po zakoupení tohoto modulu dopájet vývody. Pro práci s tímto senzorem používám knihovny Adafruit BME280 viz Adafruit knihovna na Githubu. Zdrojový kód pro pro práci s tímto modulem lze nalézt v příloze D.1. V tabulce 4.2 je znázorněno zapojení pinů BME280 modulu k Arduinu Nano.

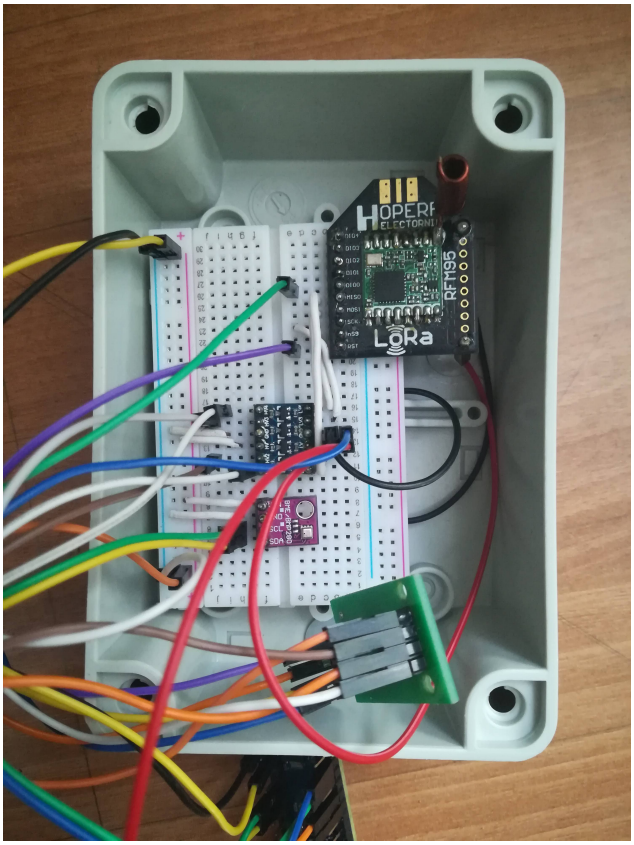
Tabulka 4.2: Zapojení pinů BME280 modulu s Arduinem Nano

BME280	Arduino
SDA	A4
SCL	A5
GND	GND
VIN	5V

4.5 Komunikační modul LoRa RFM95W pro odesílání

Arduino Nano v mém případě slouží jako sběr dat ze senzorů váhy, teploty, tlaku, vlhkosti a přeposílá tyto data přes LoRa RFM95W na WeMos D1 R2 desku, která tyto údaje příjme, rozparsruje a přes Wifi je odešle na MQTT broker server. Další informace o zvoleném MQTT broker serveru jsou

popsány v kapitole 5.1. Pro práci s modulem LoRa RFM95W jsem si stáhl knihovnu RadioHead: `http://www.airspayce.com/mikem/arduino/RadioHead/` a použil jsem ji podle tutoriálu `https://learn.adafruit.com/radio-featherwing/using-the-rfm-9x-radio` [36]. Na obrázku 4.6 lze vidět senzory s LoRa RFM95W modulem připojeným k Arduinu Nano s použitím nepájivého pole vloženého do krabičky. Tuto krabičku jsem zvolil, abych mohl zařízení provozovat ve venkovních prostorách. Krabička má stupeň krytí IP65, je tedy plně prachotěsná a vodotěsná proti tryskající vodě. Pro vyvedení drátů z krabičky by bylo vhodné vyvrtat díru a utěsnit ji. Tuto úpravu jsem však neprováděl, protože přes velikost všech drátů bych krabičku stejně neuzavřel. Nejvhodnějším řešením by bylo navrhnout desku plošného spoje a osadit do ní součástky.



Obrázek 4.6: Arduino Nano se senzory a LoRa RFM95W v krabičce

4.5.1 Napájení LoRa RFM95W modulu

RFM95W Pracuje na napájecím napětí 1,8 - 3,7 V s maximálním odběrem proudu 120 mA. Pro komunikaci používá tento modul 3,3 V logiku, proto není možné přímé propojení s Arduinem Nano, které používá 5 V. Bylo potřeba použít převodník logických úrovní z 5 V na 3,3 V. Celé Arduino napájím přes USB konektor, aby to bylo později možné napájet z powerbanky viz kapitola o externím napájení Arduina a připojených senzorů 4.2. Žádný externí zdroj pro napájení modulů jsem neměl

Tabulka 4.3: Zapojení RFM95W, převodníku logické úrovně a Arduina

RFM95W	převodník	Arduino
RST	-	D9
nSS	LV4	-
-	HV4	D10
SCK	LV3	-
-	HV3	D13
MOSI	LV2	-
-	HV2	D11
MISO	LV1	-
-	HV1	D12
DIO0	-	D2
GND	GND (LV)	-
-	GND (LV)	GND
VIN 3,3 V	LV	-
-	LV	3v3
-	HV	+5V
-	GND (HV)	GND

k dispozici, proto jsem snížil vysílací výkon RFM95W v programu na 18 dBm, aby nedošlo k větší spotřebě proudu, než kolik může Arduino Nano dodat. Arduino Nano 3.3 V pin může dodat maximální proud 100 mA.

4.5.2 Převodní deska CON RFM95

Zakoupil jsem RFM95W-868-S2R a CON RFM95 moduly. RFM95W-868-S2R je samotný vysílací a přijímací LoRa modul, který jsem připájel k desce CON RFM95. CON RFM95 je deska, která je kompatibilní k RFM95W-868-S2R modulu a lze k ní snadno připájet kolíky, které jdou zasunout do nepájivého pole.

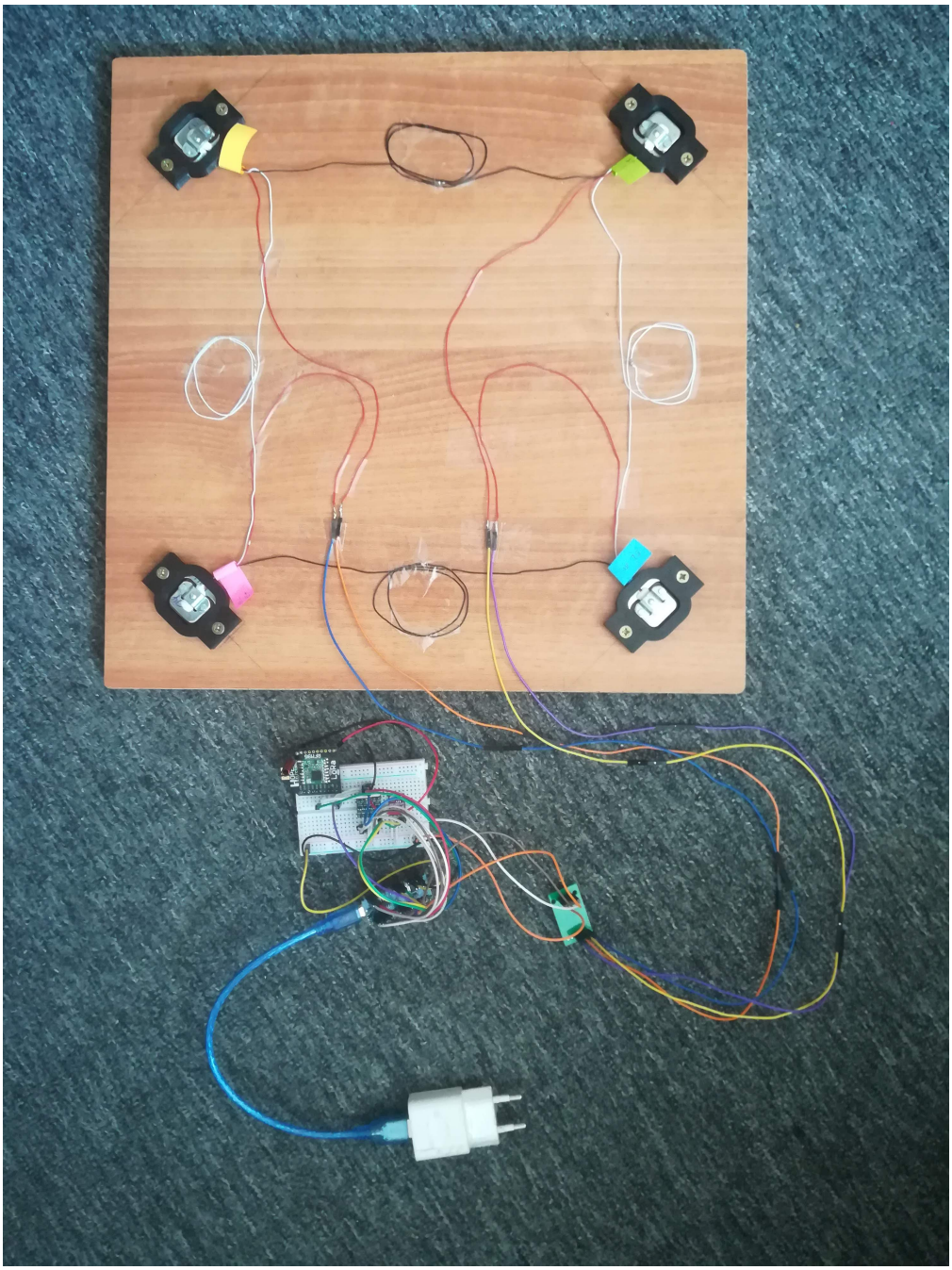
4.5.3 Tvorba 868 MHz antény

Podle videonávodu `https://www.youtube.com/watch?v=5d2GJOVMWSs` [37] jsem si vytvořil dvě spirálovité 868 MHz antény (jedna pro příjemce a druhá pro odesílatele), které jsem následně napájel na LoRa RFM95W modul na pin pro anténu.

Na anténu jsem použil měděný drát o tloušce 1 mm a obtáčel jsem ho okolo zatlučeného hřebíku o tloušce 3 mm, tak aby délka výsledné antény a počet otáček odpovídaly návodu.

Dle českého telekomunikačního úřadu je v České Republice dán rozsah kmitočtů 868–876 MHz pro bezlicenční použití a může se zde provozovat zařízení bez individuálního oprávnění. [33]

Tabulka 4.3 znázorňuje zapojení pinů LoRy RFM95W s Arduinem Nano a převodníkem logické úrovně.



Obrázek 4.4: Váha (spodní část), Arduino Nano, BME280, HX711, LoRa RFM95W, převodník logické úrovně

Kapitola 5

Návrh zařízení pro přenos dat na MQTT broker server

5.1 MQTT služba od IO Adafruit

Adafruit.com je web, který slouží jako obchod, blog, fórum, publikuje různé návody s elektronikou, vytváří knihovny pro programování a nabízí služby jako je MQTT broker server s uložištěm dat.

IO Adafruit nabízí jak placený, tak i neplacený plán MQTT broker serveru. Pro mé testování jsem využil zdarma poskytovaných služeb, které nabízí:

- 30 uložených dat za minutu
- 30-ti denní uchování dat
- 10 témat (topics, feeds)
- 5 řídicích panelů (dashboards)
- možnost si stáhnout data v csv, nebo json formátu

Pro přihlášení k odběru, nebo k zaslání zprávy na MQTT broker server se připojuje pomocí přihlašovacích údajů popsaných v tabulce 5.1.

Tabulka 5.1: [38] IO Adafruit MQTT přihlašovací detaily

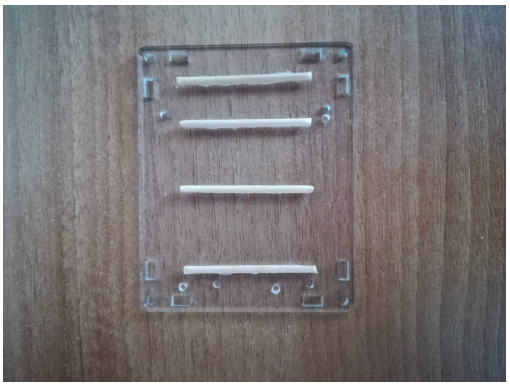
Host	io.adafruit.com
Zabezpečený (SSL) Port	8883
Nezabezpečený Port	1883
Uživatelské jméno	Adafruit IO uživatelské jméno
Heslo	Adafruit IO klíč

5.2 Vývojová deska WeMos D1 R2 UNO ESP8266

WeMos D1 R2 UNO ESP8266 deska pracuje na čipu ESP8266. Její rozložení pinů a konektorů se podobá Arduinu UNO. Tuhle desku jsem zvolil z důvodu, že obsahuje zabudovaný WiFi modul. Výhodou je, že pracuje na 3,3 V logické úrovni. Díky tomu je možné ji propojit napřímo s menším počtem drátů k ostatním modulům, protože nemusíme použít externí převodník logické úrovně. Pro WeMos desku jsem koupil transparentní krabičku viz obrázek 5.1a.

Ovšem tato krabička není zcela kompatibilní pro desku WeMos D1 R2 UNO ESP8266, proto jsem udělal několik úprav, aby pasovala. Vyvrtal jsem 3 díry do spodní plochy krabičky. Potom jsem ze přední části odpiloval část plastu okolo USB konektoru a nakonec jsem zevnitř spodní plochy krabičky přilepil špejle viz obrázek 5.1b. Použil jsem je proto, že WeMos deska zespodu není rovná, kvůli přesahujícím nožičkám od napájecího konektoru a kvůli USB portu, aby byl výš a nebránila mu přední část plastu při připojování usb kabelu. WeMos desku napájím ze sítě pomocí napájecího adaptéru s výstupem 9 V/1 A. Připojený LoRa RFM95W modul je napájený z WeMos desky z 3,3 V pinu.

(a) [39] Transparentní krabička pro Arduino UNO



(b) Spodní část krabičky s vyvrtanými dírkami a přilepenými špejlemi

Obrázek 5.1: Transparentní krabička a spodní část modifikované krabičky

5.3 Komunikační modul LoRa RFM95W pro příjem

LoRa RFM95W modul jsem připájel k převodní desce CON RFM95, jak již bylo popsáno v kapitole 4.5.2 a k tomu vytvořil 868 MHz anténu, která byla popsána v kapitole 4.5.3. Pro práci desky WeMos D1 R2 s LoRa RFM95W modulem jsem nepoužil stejnou knihovnu jako u Arduina Nano, protože se mi ji na této desce nepodařilo zprovoznit. Proto jsem použil knihovnu Arduino-LoRa od uživatele Sandeep Mistry. Odkaz na knihovnu lze najít na `https://github.com/sandeepmistry/arduino-LoRa`.

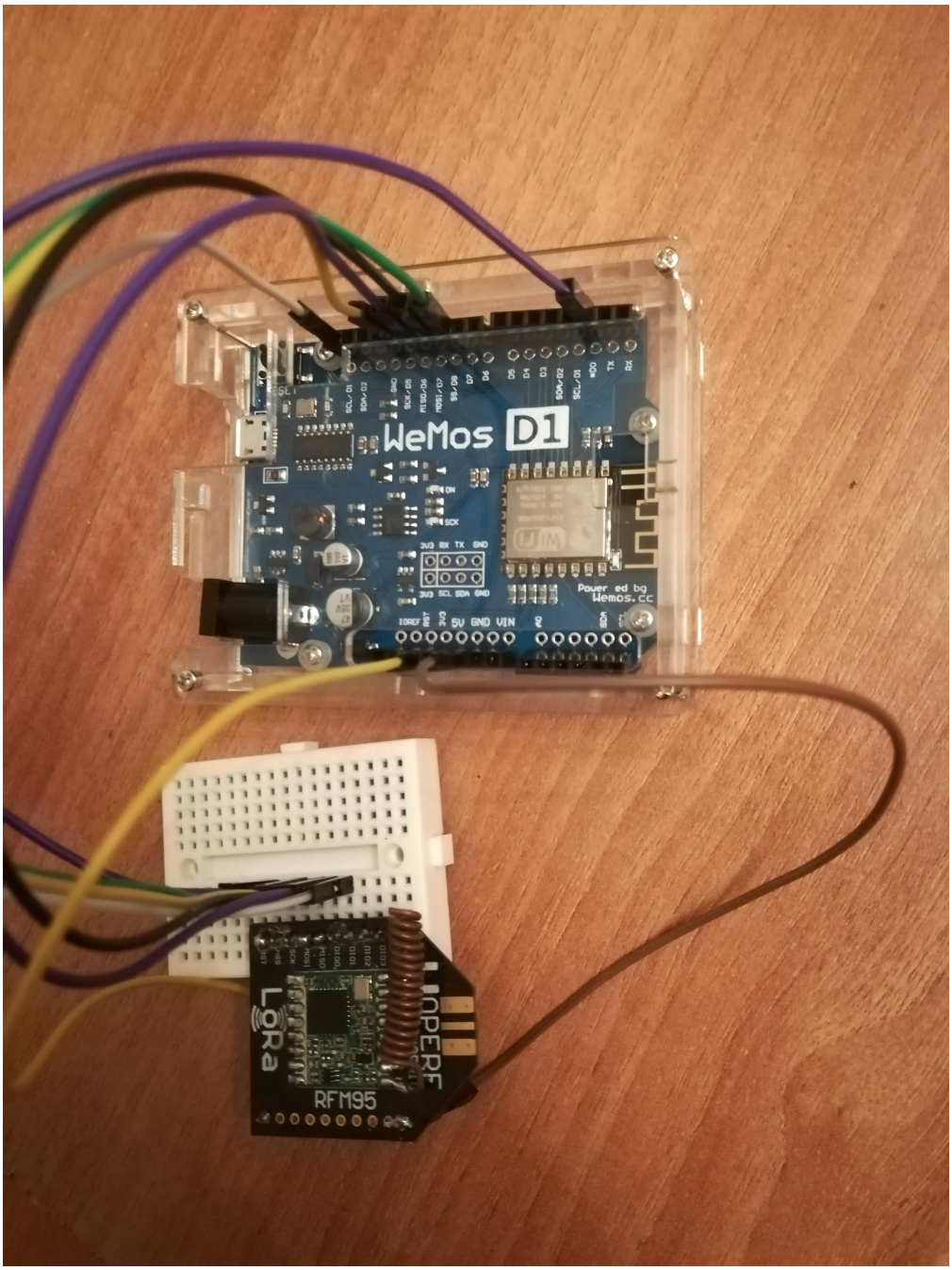
Při inicializaci LoRa RFM95W modulu ve zdrojovém kódu je velice důležité nastavit správné mapování pinů. Čísla pinů se musejí zadávat podle pinů z čipu ESP8266 a ne z desky WeMos D1 R2. Pro správné nastavení je vhodné použít například převodní tabulku, která je zachycena na obrázku 5.3. Ukázka konfigurace LoRy s mým zapojením je zobrazena v kódu 5.1. Celkový zdrojový kód je zobrazen v příloze D.6.

```
#include <LoRa.h>

const int csPin = 15;          // WeMos D8 pin - nSS ESP pin
const int resetPin = 16;       // WeMos D0 pin - RST ESP pin
const int irqPin = 5;          // WeMos D1 pin - DIO0 ESP pin

LoRa.setPins(csPin, resetPin, irqPin);
```

Listing 5.1: Inicializace Lora modulu

Wemos D1	ESP8266 Pin	Functions
D0	16	GPIO
D1	5	GPIO, **I2C** SCL
D2	4	GPIO, **I2C** SDA
D3	0	GPIO
D4	2	GPIO
D5	14	GPIO, **SPI** SCK (Serial Clock)
D6	12	GPIO, **SPI** MISO (Master in, Slave out)
D7	13	GPIO, **SPI** MOSI (Master out, Slave in)
D8	15	GPIO, **SPI** SS (Slave select)
A0	A0	Analog in, via ADC
RX	3	Receive
TX	1	Transmit

Obrázek 5.3: [40] WeMos - ESP8266 mapování pinů



Obrázek 5.2: WeMos deska v průhledné krabičce s připojeným LoRa RFM95W modulem

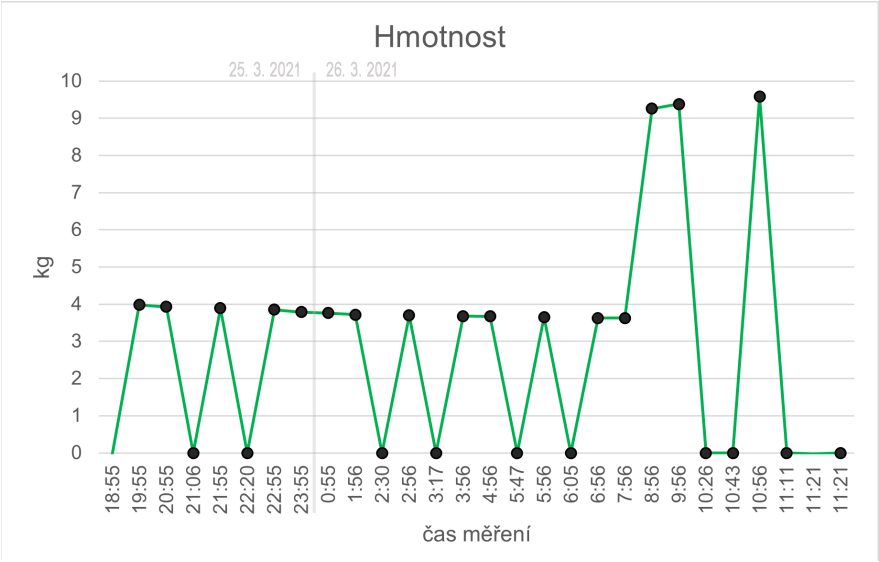
Kapitola 6

Testování, sběr dat a vizualizace v grafech

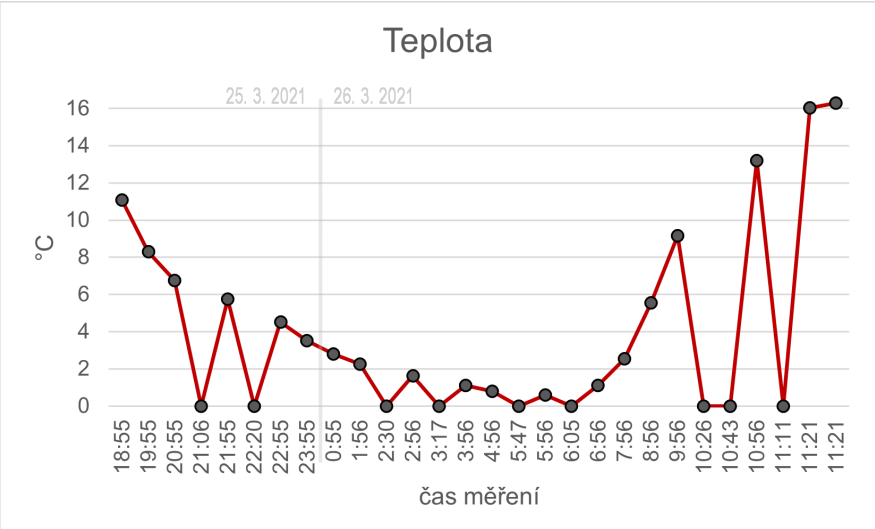
Data jsem měřil na tři části. První část probíhala od 25. 3. 2021 do 26. 3. 2021 a byla měřena s hodinovými intervaly. Do těchto dat se promítly i zachycené rádiové vlny z okolí. Tyto rádiové vlny nebyly vyslány mnou, proto se je nepodařilo rozparsovat, a proto se na MQTT broker server odeslaly hodnoty jako nulové. Nulové hodnoty na grafech hmotnosti 6.1, teploty 6.2, tlaku C.1, vlhkosti C.2 se objevují v nepravidelných časech, zatímco moje zaslaná data jsou v pravidelných hodinových intervalech.

Druhá část měření probíhala pět dní od 27. 3. 2021 do 31. 3. 2021, s tím, že měření probíhalo každou hodinu. Výsledné grafy hmotnosti C.3a, teploty C.3b, tlaku C.4a, vlhkosti C.4b jsou umístněny v příloze.

Třetí část měření probíhala 2 dny od 2. 4. 2021 do 3. 4. 2021. Hodnoty byly měřeny každých 10 minut. Veškerá naměřená data, ze kterých jsem dělal grafy jsou vložena do příloh jako tabulky.



Obrázek 6.1: Měřená hmotnost i s chybně zachycenými daty



Obrázek 6.2: Měřená teplota i s chybně zachycenými daty

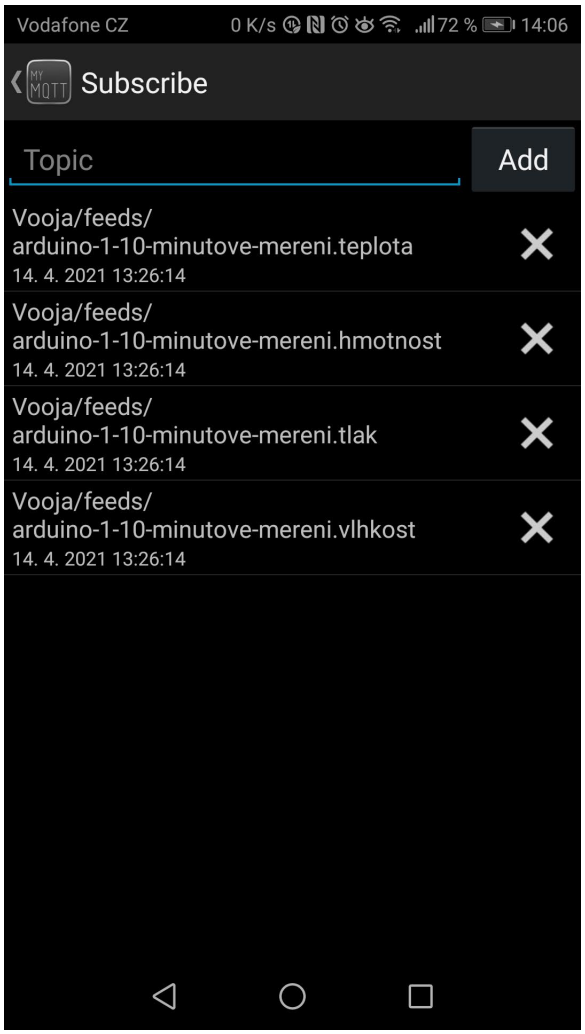
6.1 Testování a měření

Po zhotovení a úspěšném otestování funkčnosti jednotlivých částí zapojených modulů a mikropočítačů jsem přešel k měření a testování funkčnosti jako celku. WeMos desku jsem nechal umístněnou v domě a napájel ji ze sítě. Arduino Nano s váhou a senzory jsem dal na zahradu pod přístřešek vzdálený 25 metrů. LoRa signál procházel přes jednu zeď. Arduino jsem pod přístřeškem napájel taky ze sítě, ale mohlo by být napájeno i z powerbanky. Viz sekce o napájení z externího zdroje 4.2. Při těchto testování jsem již neměl propojené Arduino Nano ani WeMos mikropočítače k počítači. Proto jsem měřené hodnoty z mého notebooku sledoval přes webové rozhraní MQTT broker serveru IO Adafruitu a navíc k tomu jsem si na mobilní telefon stáhl MyMQTT aplikaci, kde jsem se připojil k mému MQTT IO Adafruit klíči a nastavil jsem si odběr témat. Přihlášení k odběru témat je možné vidět na obrázku 6.3a a příchozí zprávy na obrázku 6.3b.

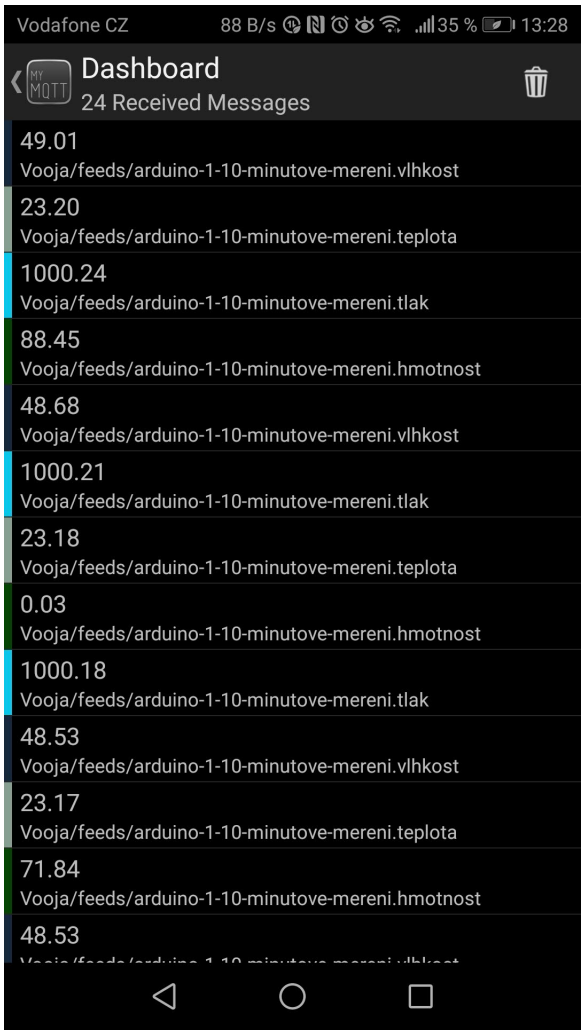
6.1.1 První měření

Arduino jsem nastavil na zhruba hodinové intervaly měření (popsáno níže). Zprávy přes LoRu jsem odesílal ve tvaru: "hmotnost teplota tlak vlhkost", kde hmotnost, teplota, tlak a vlhkost jsou naměřené hodnoty na 2 desetinná místa.

Zprávu k odeslání vytvářím ve funkci createMessage, která bere za parametr strukturu Measurements. Tato struktura je naplněna naměřenými daty. Strukturu je možné vidět v ukázce kódu 6.4, nebo jako celý kód v příloze D.2. Začátek zprávy začínám mezerou, pro usnadnění rozparsování zprávy na druhém přijímacím zařízení, protože při odesílání zprávy pomocí knihovny `RadioHead RH_RF95.h` jsou touto knihovnou na začátek každé zprávy přidány čtyři bajty sync wordu. Jelikož



(a) MQTT klientská aplikace - nastavení odběrů témat



(b) MQTT klientská aplikace, dashboard - příchozí zprávy

Obrázek 6.3: MQTT klientská aplikace, nastavení odběrů a zobrazení příchozích zpráv

na Arduinu (pro odesílání) a WeMos desce (pro přijímání) používám rozdílné knihovny pro ovládání LoRy RFM95W, tak je nutné manuálně oddělit sync word od zprávy. Vytváření zprávy je možné vidět v ukázce kódu 6.5, nebo jako celkový kód v příloze D.2.

Pro odesílání dat přes LoRu na Arduinu používám knihovnu RadioHead od Adafruit Industries. Tato knihovna, jak jsem již zmiňoval přidává před každou odesílanou zprávu čtyři bajty. Konkrétně se jedná o tyto 0xFF, 0xFF, 0x0, 0x0. Slouží jako sync word. To znamená, že pokud má odesílatel a příjemce na začátku zprávy stejný počet domluvených bajtů se stejnou hodnotu, tak se zpráva zpracuje. V opačném případě se zpráva zahazuje, protože patřila nejspíše někomu jinému, nebo je to přijatý ruch z okolních vln.

Pro příjem dat z LoRy RFM95W na desce WeMos používám knihovnu arduino-LoRa od Sandeep Mistry. Následně tato data rozděluji podle mezer. První 4 bajty sync wordu přeskakuji, poté beru postupně jednotlivé hodnoty a přeposílám (publikuji na určitý topik/téma) je na MQTT broker server IO Adafruit pomocí knihovny Adafruit MQTT Library od Adafruitu.

V případě, že by deska WeMos chtěla zpět přes LoRu odeslat potvrzovací zprávu o přijetí zprávy z Arduina, tak by musela před svou zprávu manuálně přidat dva bajty s hodnotou 0xFF (jedná se tedy o sync wordu). V opačném případě by Arduino Nano nepřijímalo pomocí knihovny RadioHead žádné zprávy od WeMos desky. Třetí bajt v poli 0x0 je null terminátor, slouží pro rozeznání konce textu. Kód je možný vidět zde 6.2.

Kontrolu doručených zpráv přes LoRu ve své práci nepoužívám z důvodu úspory odběru proudu, a tedy i výdrže baterie. Avšak v případě, že by bylo potřeba kontrolovat, zda zaslaná zpráva byla obdržena, řešení by mohlo spočívat v omezení počtu znovuzaslaných zpráv. Tímto způsobem by nedošlo v případě komplikací s deskou, nebo signálem k WeMos desce, k celodenímu, či vícedennímu neustálému zasílání zpráv od Arduina.

Způsob provedení by spočíval v tom, že Arduino Nano by po odeslání naměřených dat vyčkalo, jestli WeMos deska odpoví, pokud by neodpověděla, Arduino Nano by vyčkalo 30 sekund a opakovalo by zaslání ještě třikrát. Pokud by WeMos deska ani po třetím znovuposláním dat neodpověděla, tak by Arduino Nano další pokus o vysílání již neprováděl. Další pokus Arduina o vysílání by pokračoval za hodinu.

Avšak při tomhle způsobu by se muselo počítat s častější výměnou externí zdroje napájení, protože by se musel brát v potaz nejhorší scénář neustálého znovuzasílání zpráv, a tedy i většího odběru proudu.

Třetí řádek od spodu v kódu 6.2 je výpočet hodinového intervalu. První tři hodnoty udávají počet milisekund za hodinu. Další hodnota se odečítá kvůli některým zpožděním v kódu. Poslední část výpočtu je odečet kompenzace za vyčkávání na odeslání zpráv. Kód na WeMos desce by zůstal totožný s tímhle kódem D.6, jen by měl odkomentovaný řádek `rfm95_sendReply();`, který zasílá potvrzující zprávu o obdržení.

```
const unsigned long EXTRA_WAIT_TIME = 30000UL;
```

```
void loop()
{
    float weight = getWeight();
    Measurements mData = getMeasurements();
    mData.weight = weight;

    scale.power_down();

    rfm95_send(mData);
    Serial.println("Sent");

    int num_repeats = 0;
    while(num_repeats < 3 && !rfm95_gotReply()) {
        delay(EXTRA_WAIT_TIME);
        num_repeats++;
        rfm95_send(mData);
        Serial.println("Sent");
    }

   delay(60UL * 60UL * 1000UL - 5020UL - num_repeats * (EXTRA_WAIT_TIME +
      WAIT_TIME_FOR_REPLY)); // hodinovy interval s kompenzaci cekani na odeslani
   scale.power_up();
}

bool rfm95_gotReply()
{
  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN] = {0};
  uint8_t len;

  Serial.println("Waiting for reply ...");
  delay(10);

  if (rf95.waitAvailableTimeout(WAIT_TIME_FOR_REPLY))
  {
    // Zachyceni zpravy
    if (rf95.recv(buf, &len))
   {
      Serial.print("Got reply: ");
```

```
      Serial.println((char*)buf);

      Serial.print("RSSI: ");
      Serial.println(rf95.lastRssi(), DEC);
      return true;
    }
    else
    {
      Serial.println("Receive failed");
      return false;
    }
  }
  else
  {
    Serial.println("No reply, is there a listener around?");
    return false;
  }
}
```

Listing 6.1: Ukázka zaslání odpovědi na přijatou zprávu

```
void rfm95_sendReply()
{
    // Odeslani odpovedi
    Serial.println("Sending reply message");

    byte syncWord[3] = {0xFF, 0xFF, 0x0};
    String syncWordTex = (char*)syncWord;
    String text = syncWordTex + " Got message";

    Serial.println("Sending reply: " + text);
    delay(10);

    LoRa.beginPacket();
    LoRa.print(text);
    LoRa.endPacket();
    // Serial.println("Reply message sent");
}
```

Listing 6.2: Ukázka zaslání odpovědi na přijatou zprávu

Hodinové intervaly jsou v přibližných časových úsecích, to znamená že každý interval se bude o něco lišit v závislosti na době komunikace se senzory a časové délce odeslání dat přes LoRu. V mém případě zhruba po 105 měření po hodině, tedy 105 hodinách se celý interval posune o minutu. Pro vytvoření pauz v programu jsem použil funkci delay(ms), která bere parametr čas v milisekundách s datovým typem unsigned long. Výpočet pauzy mám jako (60 * 1000 milisekund) = jedna minuta * 60 = 1 hodina a to celé následně snižuji o konstantu 5020 ms, což je 5,02 sekund z důvodu různých zpoždění během měření a odesílání. Kód lze vidět v ukázkového kódu 6.3, nebo v kompletním kódu v příloze D.5.

```
delay(60UL * 60UL * 1000UL - 5020UL);
```

Listing 6.3: Ukázka výpočtu hodinové prodlevy v kódu

```
struct Measurements {
    float weight;         // Kg
    float temperature;    // *C
    float pressure;       // hPa
    float humidity;       // %
    float altitude;       // m
};
```

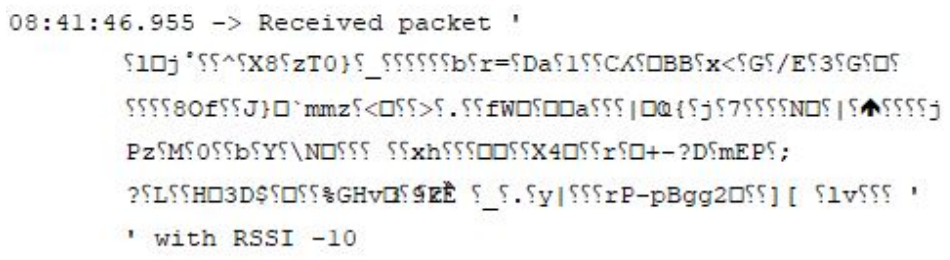
Listing 6.4: Ukázka struktury pro ukládání naměřených dat

```
String createMessage(Measurements measurements)
{
    String msg = "";
    msg += " ";
    msg += "arduino_1";       // tento radek kodu byl pridan az pri druhem mereni
    msg += " ";
    msg += measurements.weight;
    msg += " ";
    msg += measurements.temperature;
    msg += " ";
    msg += measurements.pressure;
    msg += " ";
    msg += measurements.humidity;
```

```
    return msg;
}
```

Listing 6.5: Ukázka tvorby zprávy pro odeslání

Váhu jsem měl zatíženou 16 hodin jednou cihlou, pak jsem přidal druhou a pokračoval ve sběru dat. Při měření jsem si všiml, že grafy mi ukazují na některých místech velkou výchylku, a to s hodnotu 0 u všech měření (hmotnost, teplota, tlak, vlhkost) v daný moment. Lze to vidět na grafech hmotnosti 6.1, teploty 6.2, tlaku C.1, vlhkosti C.2 u všech hodnot s nulou. Měření a zasílání hodnot na MQTT server jsem měl nastavené zhruba na každou hodinu. V grafech jsem si však všimnul, že se tam objevují i hodnoty, které nejsou v hodinovém intervalu a jsou to pouze ty nulové. Zjistil jsem, že LoRa připojená k WeMos desce zachycovala a přijímala nesouvisející rádiové signály. Tyto zprávy se nepodařilo rozparsovat, a proto se odesílaly na MQTT broker server jako nulové hodnoty. Ukázku takové zachycené zprávy z okolních rádiových vln můžete vidět na obrázku 6.4.



Obrázek 6.4: Ukázka zachycení chybné zprávy

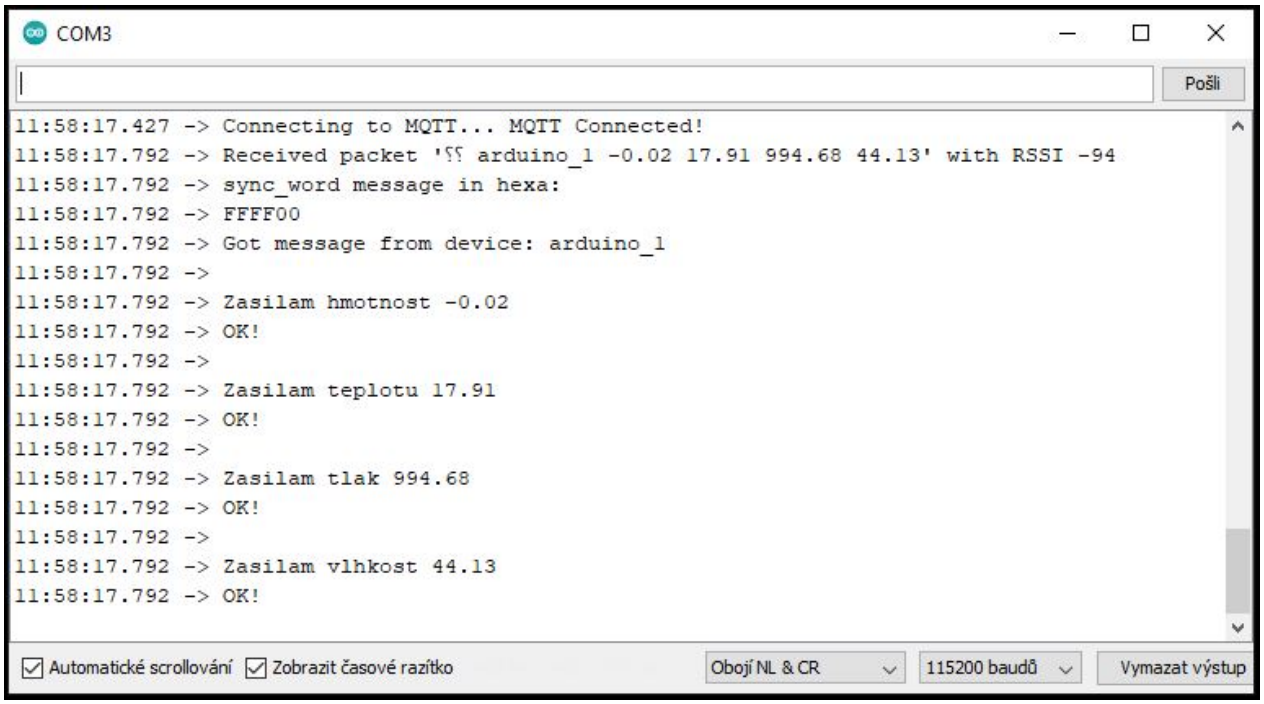
6.1.2 Druhé měření

Problém s přijímanými zachycenými rádiovými vlnami z okolí jsem vyřešil tak, že jsem v Arduinu Nano při odesílání naměřených dat přidal do zprávy ještě identifikátor Arduina. Zpráva následně vypadá takto: "`Arduino_N` hmotnost teplota tlak vlhkost", kde u `Arduino_N` "N"značí pořadové číslo arduina (k rozpoznání, které zařízení odesílalo, pokud by jich bylo více), hmotnost, teplota, tlak a vlhkost jsou naměřené hodnoty na 2 desetinná čísla. Složení zprávy v kódu lze vidět zde 6.5, nebo v kompletním kódu v příloze D.2.

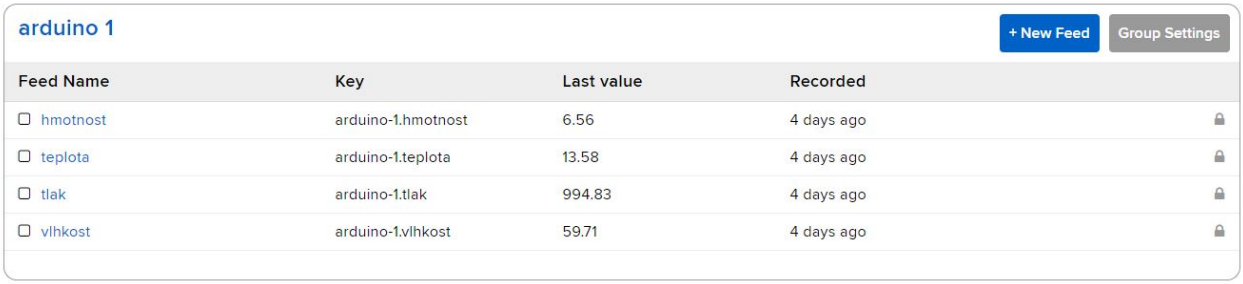
Při rozparsování takové zprávy testuji, zda obsahuje "`Arduino_1`", pokud ne, tak zprávu zahazuji. Pokud obsahuje, tak rozparsované hodnoty odesílám na MQTT broker server do skupiny topiců arduina 1. Kód lze vidět v příloze D.6. Obrázek 6.5 ukazuje zachycenou zprávu. Obrázek 6.6 ukazuje skupinu témat (topics) pro Arduino 1. V případě, že by jsme měli více arduin se senzory, tak by se vytvořily další skupiny Arduino 2, Arduino 3, ..., kam by se zasílaly data dle skupin.

Od okamžiku, kdy jsem začal testovat, zda rozparsovaná část zprávy obsahuje řetězec `Arduino_1`, se již výchylky v grafu a datech nevyskytly. Arduino Nano jsem nechal celý týden v kuse sbírat data

v hodinových intervalech a zasílal je na MQTT broker server. Data jsem pak vyobrazil z pěti dnů. Výsledné grafy lze vidět na obrázcích s hmotností C.3a, teploty C.3b, tlaku C.4a a vlhkosti C.4b.



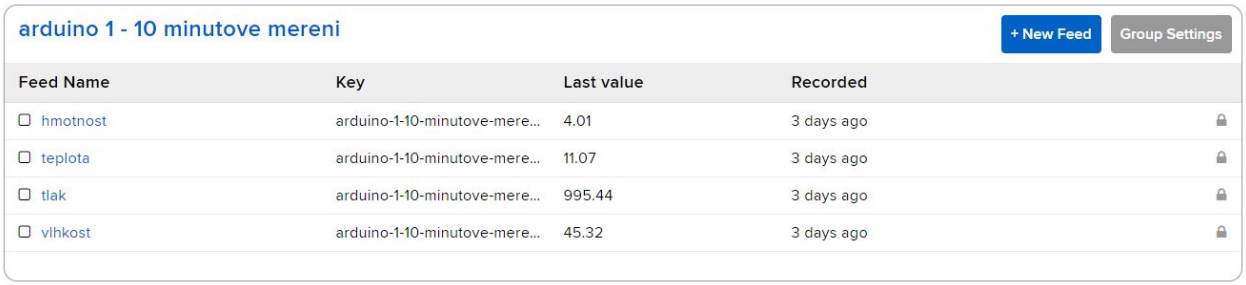
Obrázek 6.5: Konzole WeMosu při přijmu zprávy z LoRy a následné zaslání dat na MQTT broker server



Obrázek 6.6: IO Adafruit rozhraní skupin témat na MQTT broker serveru

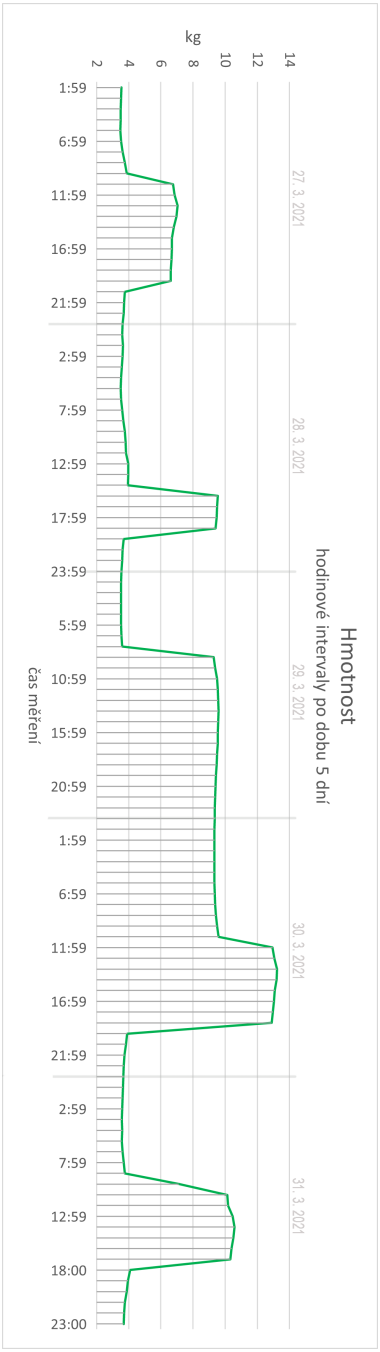
6.1.3 Třetí měření

Třetí měření trvalo dva dny. V intervalech deseti minut jsem měřil hmotnost, teplotu, tlak a vlhkost. Vytvořil jsem novou skupinu témat MQTT broker serveru, kam jsem data odesílal. Skupinu můžete vidět na obrázku 6.7. Grafy můžete vidět na obrázku hmotnosti C.5a, teploty C.5b, tlaku C.6a a vlhkosti C.6b.

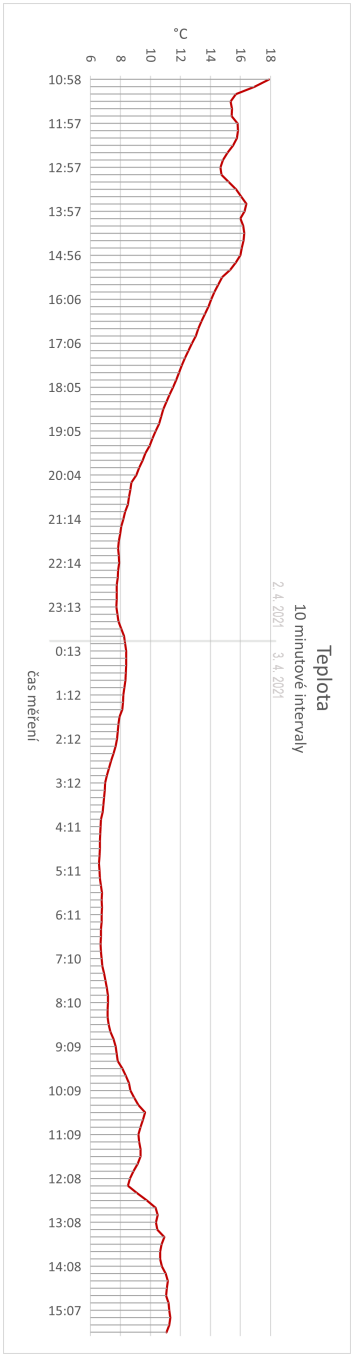


Obrázek 6.7: IO Adafruit rozhraní druhé skupin témat na MQTT broker serveru

6.2 Vizualizace naměřených hodnot



(a) Pětidenní hmotnost na váze - hodinové intervaly



(b) Dvoudenní teplota vzduchu - desetiminutové intervaly

Obrázek 6.8: Pětidenní denní graf hmotnosti a dvoudenní denní graf teploty

6.3 Zhodnocení naměřených výsledků

Data se podařilo naměřit s vypovídajícími hodnotami, jak při krátkodobém měření, tak i při dlouhodobém měření, kde zařízení byla zapnutá sedm dní v kuse. Při prvním větším měření se vyskytly problémy, kde přijímací zařízení zachytávalo i rádiové vlny, které jsem nevysílal. Tento problém, jak již bylo popsáno výše se podařilo vyřešit a další měřená data přicházela v pořádku. Při bližším pohledu na naměřená data ve grafu hmotnosti C.3a si lze všimnout, že po přiložení závaží v rozmezí od 29. 3. 8:59 do 30. 3. 10:59 naměřená váha nebyla konstantní, ale lehce se měnila v závislosti na vlivu okolí, jako je například teplota, nebo na délce trvání měření položeného závaží. Maximální váha na tomto intervalu byla 9,59 kg a minimální hodnota 9,27 kg.

Kapitola 7

Závěr

Cílem této práce byl sběr dat v pravidelných časových intervalech z monitorovaného včelího úlu, jejich přenos do mobilního telefonu pomocí MQTT protokolu a nakonec jejich vyobrazení v grafech. Vzhledem k tomu, že nemám včelí úl, zvolil jsem jako náhradní řešení různé druhy závaží převážně ve formě cihel. Cíle práce byly naplněny.

Při zhotovování této práce se vyskytlo několik problémů. Například na použitém nepájivém poli byly kontakty zdířek příliš těsné a nepodařilo se tedy do nich vsunout součástky. Při snaze toto nepájivé pole osadit kolíky, se plast okolo kolíků odtrhl. Tento problém byl vyřešen rozšířením zdířek. Další problém nastal po zapojení součástky, která po chvíli přestala fungovat. Po rozebrání celého zapojení, opětovnému sestavení a zkoumání nepájivého pole jsem zjistil, že jeden z propojů v kontaktním poli byl špatný. Při prvním delším měření byly mezi přijatými daty i chybné hodnoty. Ty byly způsobeny příjmem nesouvisejících rádiových signálů z okolí, které se nepodařilo rozparsovat a odesílaly se na MQTT broker jako nulové. Chybné zprávy se podařilo odstranit použitím identifikátoru v odesílané zprávě přes LoRu.

Do budoucna je možno provést několik rozšíření tohoto zařízení. Pro více úlů na jednom místě by bylo možné připojit váhy od každého úlu k jedné desce Arduino Nano, která by data odesílala přes LoRa vysílač na druhou desku WeMos. WeMos deska by přijatá data třídila podle úlů a odesílala by je na patřičné téma na MQTT broker. Pokud by každý včelí úl byl na jiném místě, tak by měl vlastní desku Arduino Nano s příslušnými senzory. Přenos dat by byl stejný jako v předchozím případě. Další možnost rozšíření je použití solárního panelu v kombinaci s baterií, přes kterou by bylo napájeno Arduino. Dále by bylo možné vytvořit si svoji vlastní MQTT klientskou aplikaci a zobrazovat si tak přehledně data podle svého vkusu.

Literatura

1. ĎAĎO, Stanislav; KREIDL, Marcel. *SENZORY a měřící obvody*. 1. vyd. Česká Republika: ČVUT, 1996. ISBN 80-01-01500-9.
2. SCHAFFEROVÁ, Magdalena. *Senzor, detektor, snímač a jiné zapeklité pojmy. Zooco: Vlog Zooco | Žijte chytře* [online]. 2018 [cit. 2021-03-14]. Dostupné z: `https://www.zooco.io/blog/senzor-detektor-snimac-jine-zapeklite-pojmy/`.
3. MILOŠ, Kmínek. *Měření teploty: Ústavu počítačové a řídicí techniky. MĚŘICÍ A ŘÍDICÍ TECHNIKA* [online]. 2005 [cit. 2021-03-14]. Dostupné z: `http://uprt.vscht.cz/kminekm/mrt/F4/F4k43-tepl.htm`.
4. MAURO, Marco. *OpenQCM the Temperature Sensor Using a Thermistor with Arduino: Quartz Crystal Microbalance* [online]. 2015 [cit. 2021-03-17]. Dostupné z: `https://openqcm.com/openqcm-temperature-sensor-using-a-thermistor-with-arduino.html`.
5. SHZR. *How to calculate temperature through NTC thermistor without its datasheet? Explore our Questions: Electrical Engineering Stack Exchange* [online]. 2017 [cit. 2021-03-17]. Dostupné z: `https://electronics.stackexchange.com/questions/323043/how-to-calculate-temperature-through-ntc-thermistor-without-its-datasheet`.
6. PETR, Danihelka; RADANA, Bodláková; JOSEF, Donát; FRANTIŠEK, Kocina; BLANKA, Kouřilová; MILENA, Luxíková; MARKÉTA, Melicharová; STANISLAV, Votýpka; HANA, Železná. *ELEKTROTECHNICKÝ ZÁKLAD* [online]. 1. vyd. Česká Republika, 2011 [cit. 2021-04-01]. Dostupné z: `https://www.sosehl.cz/phprs/download/ucebnice/Elektrotechnicky_zaklad_ucebni_text.pdf`.
7. *Dělič napětí: ELUC* [online] [cit. 2021-04-01]. Dostupné z: `https://eluc.kr-olomoucky.cz/verejne/lekce/436`.
8. *Interface BME280 Temperature, Humidity & Pressure Sensor with Arduino: Last Minute Engineers* [online]. 2021 [cit. 2021-03-17]. Dostupné z: `https://lastminuteengineers.com/bme280-arduino-tutorial/`.

9. SMOLKA, Václav. *Měření a zaznamenávání tlaku vzduchu: Předpověď počasí* [online]. 2015 [cit. 2021-04-14]. Dostupné z: https://www.in-pocasi.cz/clanky/teorie/tlak-vzduchu-4.8.2015/.

10. ŠUSTROVÁ, Nela. *OBJEM, HUSTOTA A TLAK, ARCHIMEDŮV A PASCALŮV ZÁKON: Inkluzivní škola* [online] [cit. 2021-04-14]. Dostupné z: https://www.inkluzivniskola.cz/sites/default/files/uploaded/f_objemhustotatlak_a_arch.pasc_.zakony.pdf.

11. GASSMANN, Eugen; GRIES, Anna. *Electronic Pressure Measurement: Basics, applications and instrument selection* [online]. 1. vyd. Germany: Sellier Druck GmbH, 2010 [cit. 2021-04-09]. Dostupné z: https://microsites.wika.com/upload/Handbook_ElectronicPressure_en_co_34154.pdf.

12. *Wheatstone Bridge: Basic Electronics Tutorials and Revision* [online]. 2021 [cit. 2021-04-15]. Dostupné z: https://www.electronics-tutorials.ws/blog/wheatstone-bridge.html.

13. KADLEC, Karel; KMÍNEK, Miloš. *MĚŘICÍ A ŘÍDICÍ TECHNIKA: Ústav počítačové a řídicí techniky* [online]. 2005 [cit. 2021-04-23]. Dostupné z: http://uprt.vscht.cz/ucebnice/mrt/F4/F4k42-tlak.htm.

14. VOJÁČEK, Antonín. *Odporové tenzometry - princip, provedení, použití, historie: automatizace.hw.cz rady a poslední novinky z oboru* [online]. 2017 [cit. 2021-04-27]. Dostupné z: https://automatizace.hw.cz/foliove-odporove-tenzometry-princip-provedeni-pouziti-historie.html.

15. AL-MUTLAQ, Sarah. *Getting Started with Load Cells: sparkfun start something* [online]. 2015 [cit. 2021-04-26]. Dostupné z: https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all.

16. PELAYO, Roland. *How to Use Load Cell with HX711 and Arduino: Teach Me Microcontrollers* [online]. 2021 [cit. 2021-04-26]. Dostupné z: https://www.teachmemicro.com/how-to-use-load-cell-hx711-arduino/.

17. *analogRead(): Arduino* [online]. 2021 [cit. 2021-04-26]. Dostupné z: https://www.arduino.cc/en/Reference.AnalogRead.

18. PELAYO, Roland. *Arduino Sensor Interfacing Tutorial: Teach Me Microcontrollers* [online]. 2021 [cit. 2021-04-26]. Dostupné z: https://www.teachmemicro.com/arduino-sensor-tutorial/.

19. BROWN, Robert. *CONNECTING HX711 LOAD CELL TO ARDUINO: NerdyTechy - Technologies, Gadgets, Circuits Reviews and Tutorials for Hobbyists* [online]. 2020 [cit. 2021-04-26]. Dostupné z: https://nerdytechy.com/connecting-hx711-load-cell-to-arduino/.

20. GUDINO, Miguel. *Engineering Resources: Basics of Analog-to-Digital Converters: Arrow Electronics* [online]. 2018 [cit. 2021-04-26]. Dostupné z: https://www.arrow.com/en/research-and-events/articles/engineering-resource-basics-of-analog-to-digital-converters.

21. *Sensors and Transducers: Basic Electronics Tutorials and Revision* [online]. 2021 [cit. 2021-04-26]. Dostupné z: https://www.electronics-tutorials.ws/io/io_1.html.

22. MALÝ, Martin. *Hradla, volty, jednočipy Úvod do bastlení*. 1. vyd. Česká Republika: CZ.NIC, 2017. ISBN 978-80-88168-26-3.

23. MALÝ, Martin. *Protokol MQTT: komunikační standard pro IoT: Root.cz* [online]. 2016 [cit. 2021-04-21]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/.

24. BERNSTEIN, Corinne; BRUSH, Kate; GILLIS, Alexander S. *MQTT (MQ Telemetry Transport): IoTAgenda.com* [online]. 2021 [cit. 2021-04-21]. Dostupné z: https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport.

25. PRAUZEK, Michal. *ČÍSLICOVÁ A MIKROPROCESOROVÁ TECHNIKA* [online]. 1. vyd. Česká Republika: VŠB, 2013 [cit. 2021-04-20]. Dostupné z: https://homel.vsb.cz/~pra132/files/CMT_prauzek_final_1_2.pdf.

26. GUPTA, Neeraj. *NETWORKING AND COMMUNICATION: Neeraj Gupta portfolio* [online]. 2020 [cit. 2021-04-21]. Dostupné z: http://fabacademy.org/2020/labs/akgec/students/neeraj-gupta/week15.html.

27. DUDÁČEK, Karel. *Sériová rozhraní SPI, Microwire, I2C a CAN: Sériová rozhraní a sběrnice* [online]. 2002 [cit. 2021-04-18]. Dostupné z: http://home.zcu.cz/~dudacek/NMS/Seriova_rozhrani.pdf.

28. WEIXIANG, Chen. *SPI: Serial Peripheral Interface: Hardware* [online] [cit. 2021-04-18]. Dostupné z: http://chenweixiang.github.io/assets/SPI_three_slaves.png.

29. *USB kabel: Zapojení kabelu - Jak správně zapojit ...* [Online]. 2015 [cit. 2021-04-21]. Dostupné z: http://zapojenikabelu.cz/usb.html.

30. ŘEHÁK, Jan. *USB - Universal Serial Bus - Popis rozhraní: vyvoj.hw* [online]. 2002 [cit. 2021-04-21]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/rozhrani/usb/usb-universal-serial-bus-popis-rozhrani.html.

31. *K PC lze připojit až 127 USB zařízení: Chip.cz - recenze a testy* [online]. 2011 [cit. 2021-04-21]. Dostupné z: https://www.chip.cz/casopis-chip/earchiv/vydani/r-2011/az-127-usb/.

32. SHAW, Keith. *What is Wi-Fi and why is it so important?: NETWORKWORLD FROM IDG* [online]. 2020 [cit. 2021-04-22]. Dostupné z: https://www.networkworld.com/article/3560993/what-is-wi-fi-and-why-is-it-so-important.html.

33. *Využívání vymezených rádiových kmitočtů: Český telekomunikační úřad* [online]. 2018 [cit. 2021-04-22]. Dostupné z: https://www.ctu.cz/vyuzivani-vymezenych-radiovych-kmitoctu.

34. DOSTÁLEK, Libor; KABELOVÁ, Alena. *Velký průvodce protokoly TCP/IP a systémem DNS*. 5. vyd. Česká Republika: Computer Press, 2012. ISBN 978-80-251-2236-5.

35. LUUK, Indrek. *Arduino Bathroom Scale With 50 Kg Load Cells and HX711 Amplifier: How-To Guides - Circuit Journal* [online] [cit. 2021-04-09]. Dostupné z: https://circuitjournal.com/50kg-load-cells-with-HX711.

36. ADA, lady. *Using the RFM9X Radio: Adafruit learning* [online]. 2021 [cit. 2021-04-01]. Dostupné z: https://learn.adafruit.com/radio-featherwing/using-the-rfm-9x-radio.

37. LIE, Robert. *How to create a copper 868MHz coil antenna*. 2017. Dostupné také z: https://www.youtube.com/watch?v=5d2GJOVMWSs.

38. *MQTT Connection Details: Adafruit IO MQTT API* [online] [cit. 2021-04-14]. Dostupné z: https://io.adafruit.com/api/docs/mqtt.html#mqtt-connection-details.

39. *Transparentní krabička pro Arduino UNO: ASKARDUINO* [online]. 2021 [cit. 2021-03-20]. Dostupné z: https://www.laskarduino.cz/transparentni-krabicka-pro-arduino-uno/.

40. FITZPATRICK, Martin. *Wemos D1 pin numbers & functions: Pin mapping and I2C/SPI from MicroPython* [online]. 2020 [cit. 2021-03-21]. Dostupné z: https://www.mfitzp.com/article/wemos-d1-pins-micropython/.

SEZNAM PŘÍLOH

Příloha A

Velké tabulky

Tabulka A.1: Přehled teploměrů

skupina teploměrů	typ teploměru	fyzikální princip
dilatační teploměry	plynový tenzní kapalinový kovový	změna tlaku změna tenze par změna objemu délková roztažnost
elektrické teploměry	termoelektrické odporové kovové odporové polovodičové, diodové	termoelektrický jev změna elektrického odporu změna prahového napětí
speciální teploměry	keramické žároměrky teploměrná tělíska teploměrné barvy	bod měknutí bod tání změna barvy
bezdotykové teploměry	širokopásmové pyrometry monokrystalické pyrometry poměrové pyrometry termovize	zachycení veškerého teplotního záření zachycení úzkého svazku teplotního záření srovnání dvou svazků teplotního záření o různých vlnových délkách snímání teplotního obrazu tělesa

Příloha B

Tabulky s naměřenými daty

Tabulka B.1: Jednodenní data se zachycenými vlnami

čas (HH:MM)	**hmotnost (kg)**	**teplota (°C)**	**tlak (hPa)**	**vlhkost (%)**
19:55	3.98	8.29	999.08	63.41
20:55	3.93	6.74	999.68	67.67
21:06	0	0	0	0
21:55	3.89	5.75	999.98	70.2
22:20	0	0	0	0
22:55	3.85	4.53	1000.08	73.62
23:55	3.79	3.52	1000.27	75.69
0:55	3.76	2.79	1000.54	77.91
1:56	3.72	2.26	1000.63	78.61
2:30	0	0	0	0
2:56	3.70	1.62	1000.65	79.58
3:17	0	0	0	0
3:56	3.68	1.12	1000.84	80.96
4:56	3.67	0.78	1000.94	80.83
5:47	0	0	0	0
5:56	3.65	0.58	1001.01	82.02
6:05	0	0	0	0
6:56	3.63	1.1	1001.38	81.58
7:56	3.63	2.54	1001.56	80.65
8:56	9.26	5.54	1001.51	76.29
9:56	9.38	9.17	1001.78	64.68
10:26	0	0	0	0

10:43	0	0	0	0
10:56	9.59	13.21	1001.62	51.9
11:11	0	0	0	0
11:21	-0.02	16.03	1001.28	70.68
11:21	0	16.3	1001.24	69.22

Tabulka B.2: Pětidenní data s hodinovými intervaly

čas (HH:MM)	**hmotnost (kg)**	**teplota (°C)**	**tlak (hPa)**	**vlhkost (%)**
2:59	3.52	4.13	997.45	71.24
3:59	3.49	3.67	997.41	73.18
4:59	3.49	3.64	997.43	73.44
5:59	3.48	3.31	997.62	75.03
6:59	3.53	4.42	997.57	75.18
7:59	3.62	6.73	997.51	72.48
8:59	3.77	10.20	997.63	53.15
9:59	3.87	12.76	997.89	48.62
10:59	6.75	14.58	997.94	42.36
11:59	6.86	16.98	998.14	37.08
12:59	7.03	19.84	998.47	32.62
13:59	6.97	19.77	998.90	33.25
14:59	6.81	16.86	999.73	42.04
15:59	6.69	14.33	1000.61	56.93
16:59	6.69	13.08	1000.91	59.01
17:59	6.67	12.26	1001.69	60.48
18:59	6.62	11.31	1002.99	43.77
19:59	6.62	10.88	1003.40	45.66
20:59	3.75	10.21	1004.01	47.80
21:59	3.71	8.85	1005.11	50.58
22:59	3.68	7.85	1005.88	54.88
23:59	3.63	6.51	1006.31	59.21
0:59	3.60	5.40	1007.34	64.42
1:59	3.64	6.02	1007.28	64.64
2:59	3.63	5.87	1007.44	64.36
3:59	3.58	4.97	1008.21	66.87
4:59	3.52	3.72	1008.59	68.96

5:59	3.49	2.69	1009.09	72.33
6:59	3.53	3.32	1009.77	73.19
7:59	3.60	4.77	1010.39	70.02
8:59	3.66	6.57	1010.73	62.61
9:59	3.75	9.27	1010.87	51.65
10:59	3.80	11.71	1010.51	44.43
11:59	3.83	13.48	1010.03	36.26
12:59	3.98	15.00	1009.72	32.33
13:59	3.98	14.92	1009.28	33.00
14:59	3.95	14.14	1008.87	34.61
15:59	9.54	13.88	1008.60	34.28
16:59	9.50	12.81	1008.44	35.78
17:59	9.47	11.84	1008.35	41.51
18:59	9.40	9.75	1008.72	48.17
20:59	3.68	6.38	1009.42	58.81
21:59	3.63	5.16	1009.75	62.42
22:59	3.59	4.29	1010.02	65.51
23:59	3.55	3.42	1010.09	67.38
1:59	3.53	3.10	1010.16	71.52
2:59	3.52	2.90	1010.00	73.27
3:59	3.52	3.10	1010.01	72.97
4:59	3.52	3.33	1009.96	73.45
5:59	3.53	3.70	1010.11	73.96
6:59	3.55	4.35	1010.55	72.90
7:59	3.60	5.55	1010.63	68.45
8:59	9.27	7.70	1010.87	62.29
9:59	9.37	10.09	1010.70	53.42
10:59	9.50	13.05	1010.42	47.66
11:59	9.53	15.27	1010.09	41.13
12:59	9.56	15.45	1009.64	40.38
13:59	9.58	15.29	1009.52	42.31
14:59	9.57	15.04	1009.26	42.29
15:59	9.55	14.68	1009.17	46.37
16:59	9.53	14.09	1009.16	50.97
17:59	9.50	13.36	1008.94	54.96
18:59	9.46	12.51	1009.17	59.07

19:59	9.43	11.76	1009.50	64.41
20:59	9.40	11.25	1009.70	69.13
21:59	9.38	10.87	1009.64	73.74
22:59	9.36	10.52	1009.58	75.50
23:59	9.35	10.37	1009.19	76.56
0:59	9.34	10.20	1009.34	77.32
1:59	9.34	10.13	1009.28	78.08
2:59	9.34	9.98	1009.32	78.02
3:59	9.34	9.91	1009.00	78.57
4:59	9.34	9.78	1009.01	78.94
5:59	9.34	9.59	1009.22	79.64
6:59	9.36	9.70	1009.59	80.11
7:59	9.38	10.18	1009.94	79.34
8:59	9.43	11.37	1010.00	77.10
9:59	9.50	13.78	1010.06	67.15
10:59	9.59	17.06	1009.75	58.46
11:59	12.95	20.12	1009.12	47.03
12:59	13.06	22.79	1008.40	36.78
13:59	13.22	24.37	1007.45	34.38
14:59	13.19	23.62	1007.14	34.72
15:59	13.09	21.71	1006.82	38.37
16:59	13.03	20.56	1006.43	41.12
17:59	12.97	19.55	1006.42	44.45
18:59	12.89	18.21	1006.66	52.68
19:59	3.91	16.18	1007.00	57.10
20:59	3.82	14.24	1007.05	61.55
21:59	3.74	12.70	1006.98	63.97
22:59	3.69	11.29	1006.86	66.42
23:59	3.67	10.56	1006.99	67.97
0:59	3.64	9.70	1006.70	70.13
1:59	3.61	8.85	1006.59	72.05
2:59	3.60	8.19	1006.46	74.29
3:59	3.58	7.72	1006.00	76.51
4:59	3.60	7.68	1006.01	78.04
5:59	3.56	7.09	1006.26	78.17
6:59	3.61	7.76	1006.17	80.97

7:59	3.68	9.38	1005.84	80.96
8:59	3.77	11.92	1005.58	76.58
9:59	7.13	15.46	1004.92	66.03
10:59	10.13	19.32	1004.36	52.85
11:59	10.19	22.51	1003.66	40.70
12:59	10.46	25.05	1002.89	35.48
14:00	10.58	25.82	1002.02	32.91
15:00	10.50	25.32	1001.21	32.54
16:00	10.40	24.21	1000.55	33.09
17:00	10.32	23.15	1000.04	36.27
18:00	4.09	21.80	999.69	40.24
19:00	3.96	19.64	999.73	48.11
20:00	3.87	17.19	999.83	54.01
21:00	3.76	14.96	999.59	59.13
22:00	3.72	13.53	999.34	63.15
23:00	3.68	12.30	999.14	66.91

Tabulka B.3: Dvoudenní data s desetiminutovými intervaly

čas (HH:MM)	**hmotnost (kg)**	**teplota (°C)**	**tlak (hPa)**	**vlhkost (%)**
11:08	-0.03	16.93	994.76	43.94
11:18	-0.05	15.70	994.73	48.19
11:28	-0.04	15.35	994.73	49.83
11:38	-0.07	15.43	994.63	49.38
11:48	-0.08	15.42	994.52	48.12
11:57	-0.27	15.82	994.54	47.45
12:07	-0.24	15.84	994.55	46.79
12:17	-0.18	15.76	994.56	46.06
12:27	-0.15	15.51	994.53	45.92
12:37	-0.10	15.14	994.48	46.55
12:47	-0.08	14.84	994.34	47.36
12:57	-0.06	14.67	994.30	47.75
13:07	4.00	14.75	994.14	49.12
13:17	04.02	15.24	993.98	49.39
13:27	04.03	15.71	993.98	47.12
13:37	04.04	16.05	993.90	46.60

13:47	04.07	16.39	993.78	45.25
13:57	04.09	16.29	993.68	44.12
14:07	04.09	16.00	993.51	45.12
14:17	4.10	16.18	993.46	46.03
14:27	04.09	16.26	993.44	44.62
14:36	04.09	16.20	993.31	44.51
14:46	04.08	16.09	993.27	45.25
14:56	04.08	15.99	993.32	44.66
15:06	04.06	15.69	993.29	45.12
15:16	04.07	15.33	993.32	45.88
15:36	04.06	14.79	993.16	46.64
15:46	8.87	14.52	993.03	47.04
15:56	8.87	14.28	992.93	47.33
16:06	8.87	14.07	992.86	47.52
16:16	8.84	13.87	992.92	47.01
16:26	8.85	13.65	993.00	46.40
16:36	8.84	13.40	992.85	46.94
16:46	8.85	13.19	992.77	47.24
16:56	8.85	13.03	992.68	46.62
17:06	8.85	12.77	992.68	47.77
17:15	8.85	12.54	992.74	47.99
17:25	8.85	12.31	992.77	47.36
17:35	8.86	12.09	992.62	48.36
17:45	8.86	11.92	992.65	48.70
17:55	8.86	11.72	992.63	48.49
18:05	8.86	11.51	992.67	48.70
18:15	8.87	11.29	992.65	49.06
18:25	8.87	11.07	992.60	49.56
18:35	8.87	10.87	992.66	49.72
18:45	04.06	10.71	992.62	49.95
18:55	04.05	10.58	992.68	50.54
19:05	04.04	10.35	992.67	50.86
19:15	04.06	10.14	992.80	51.24
19:25	04.04	9.95	992.87	52.09
19:35	04.03	9.66	992.91	52.35
19:45	04.01	9.48	992.79	53.24

19:54	04.01	9.25	992.91	54.17
20:04	04.01	09.05	993.05	54.57
20:24	3.99	8.74	992.86	55.50
20:34	3.98	8.67	992.86	55.62
20:44	3.99	8.57	992.87	56.56
20:54	3.99	8.50	993.09	57.08
21:04	3.96	8.32	993.02	56.83
21:14	3.96	8.20	993.00	57.91
21:24	3.95	08.06	993.15	58.88
21:34	3.94	7.98	993.18	59.42
21:44	3.93	7.89	993.19	59.79
21:54	3.94	7.84	993.40	60.53
22:04	3.94	7.89	993.21	59.96
22:14	3.92	7.91	993.19	59.66
22:24	3.92	7.84	993.06	60.22
22:33	3.92	7.81	992.98	59.73
22:43	3.92	7.76	992.95	60.32
22:53	3.92	7.74	993.05	60.80
23:03	3.92	7.74	993.07	60.16
23:13	3.92	7.73	993.22	60.60
23:23	3.92	7.80	993.15	60.59
23:33	3.91	7.87	993.10	60.66
23:43	3.91	08.06	993.03	61.72
23:53	3.91	8.23	993.00	62.25
0:03	3.91	8.32	993.00	62.12
0:13	3.91	8.37	992.99	61.36
0:23	3.91	8.39	992.91	60.70
0:33	3.91	8.37	992.89	60.64
0:43	3.91	8.35	992.82	60.59
0:53	3.90	8.33	992.73	60.22
1:03	3.90	8.26	992.63	60.61
1:12	3.90	8.19	992.52	60.74
1:22	3.89	8.18	992.37	60.84
1:32	3.89	8.12	992.46	60.57
1:42	3.90	7.93	992.49	60.95
1:52	3.89	7.87	992.53	61.39

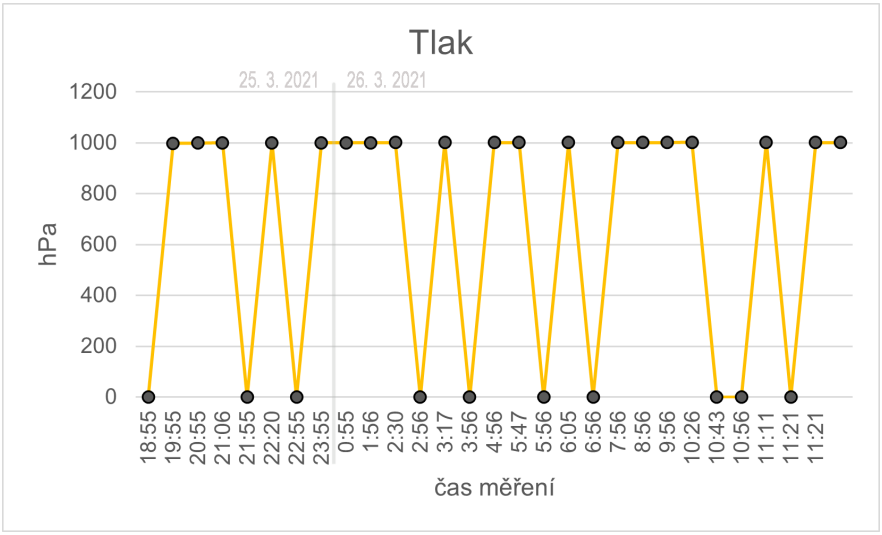
2:02	3.89	7.81	992.40	62.19
2:12	3.89	7.77	992.31	61.91
2:22	3.88	7.67	992.35	61.79
2:32	3.88	7.54	992.37	62.12
2:42	3.88	7.37	992.40	62.42
2:52	3.88	7.24	992.45	62.53
3:02	3.88	07.09	992.64	62.88
3:12	3.88	6.99	992.55	63.78
3:22	3.87	6.95	992.58	64.55
3:32	3.87	6.91	992.51	64.27
3:41	3.86	6.86	992.46	64.67
3:51	3.86	6.81	992.37	64.62
4:01	3.86	6.71	992.33	64.59
4:11	3.85	6.67	992.31	64.62
4:21	3.85	6.66	992.28	64.73
4:31	3.84	6.63	992.20	64.38
4:41	3.84	6.63	992.22	64.39
4:51	3.84	6.60	992.27	64.11
5:01	3.84	6.57	992.27	63.97
5:11	3.84	6.61	992.34	64.02
5:21	3.84	6.64	992.42	64.37
5:31	3.84	6.70	992.61	64.43
5:41	3.84	6.76	992.80	64.32
5:51	3.84	6.75	992.88	64.31
6:01	3.84	6.76	993.03	65.38
6:11	3.84	6.75	993.20	65.64
6:20	3.84	6.75	993.28	65.92
6:30	3.84	6.71	993.32	65.93
6:40	3.83	6.70	993.46	65.37
6:50	3.84	6.68	993.57	65.19
7:00	3.83	6.69	993.67	65.30
7:10	3.84	6.75	993.76	65.49
7:20	3.84	6.80	993.82	65.45
7:30	3.83	6.90	993.86	64.83
7:40	3.83	07.01	993.84	63.72
7:50	3.83	07.09	993.95	63.91

8:00	3.83	7.16	994.03	63.35
8:10	3.83	7.16	994.08	61.90
8:20	3.83	7.14	994.19	64.15
8:30	3.83	7.14	994.26	67.03
8:40	3.84	7.22	994.33	67.65
8:50	3.83	7.34	994.37	67.86
8:59	3.83	7.54	994.39	66.53
9:09	3.83	7.68	994.40	66.08
9:19	3.83	7.75	994.37	65.21
9:29	3.85	7.81	994.43	65.83
9:39	3.84	8.13	994.59	66.09
9:49	3.84	8.36	994.60	64.54
9:59	3.84	8.58	994.53	63.45
10:09	3.84	8.66	994.57	61.80
10:19	3.86	8.92	994.52	62.50
10:29	3.86	9.19	994.49	60.36
10:39	3.87	9.64	994.51	59.14
10:49	3.88	9.51	994.69	57.73
10:59	3.89	9.35	994.67	57.46
11:09	3.90	9.19	994.58	57.46
11:19	3.89	9.25	994.63	57.06
11:29	3.87	9.33	994.92	58.62
11:38	3.88	9.35	995.02	61.87
11:48	3.87	9.16	995.19	59.46
11:58	3.88	8.88	995.47	59.80
12:08	3.90	8.65	995.68	62.70
12:18	3.90	8.49	995.57	64.76
12:28	3.57	9.10	995.43	65.29
12:38	3.24	9.75	995.25	63.44
12:48	2.99	10.34	995.14	58.62
12:58	3.46	10.49	995.26	51.63
13:08	3.74	10.37	995.21	49.17
13:18	3.89	10.46	995.20	50.07
13:28	3.89	10.94	995.24	47.89
13:38	3.93	10.75	995.40	45.89
13:48	3.94	10.64	995.41	47.50

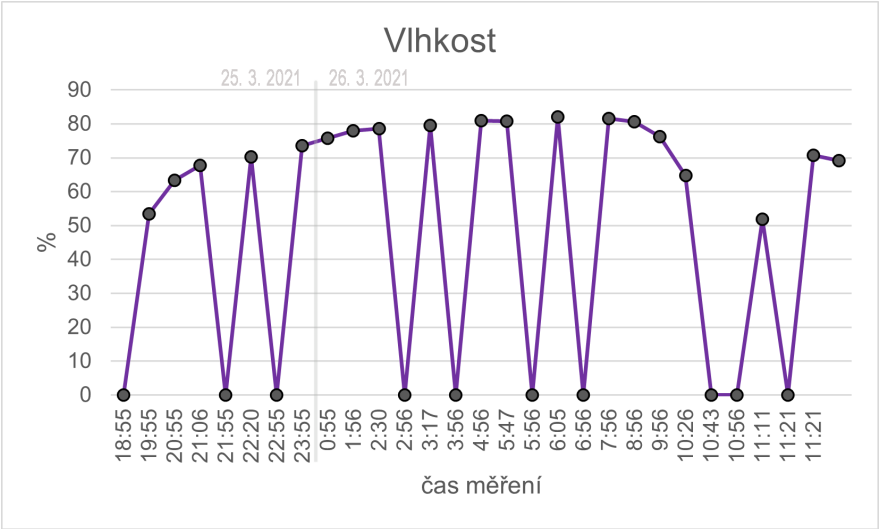
13:58	3.94	10.64	995.49	48.77
14:08	3.92	10.76	995.48	48.49
14:17	3.94	11.02	995.34	50.60
14:27	3.95	11.17	995.25	48.58
14:37	3.97	11.10	995.29	47.37
14:47	3.98	11.05	995.41	46.79
14:57	3.98	11.21	995.17	49.86
15:07	3.98	11.26	995.17	46.12
15:17	04.01	11.32	995.18	45.87
15:27	04.01	11.25	995.25	44.31
15:37	04.01	11.07	995.44	45.32

Příloha C

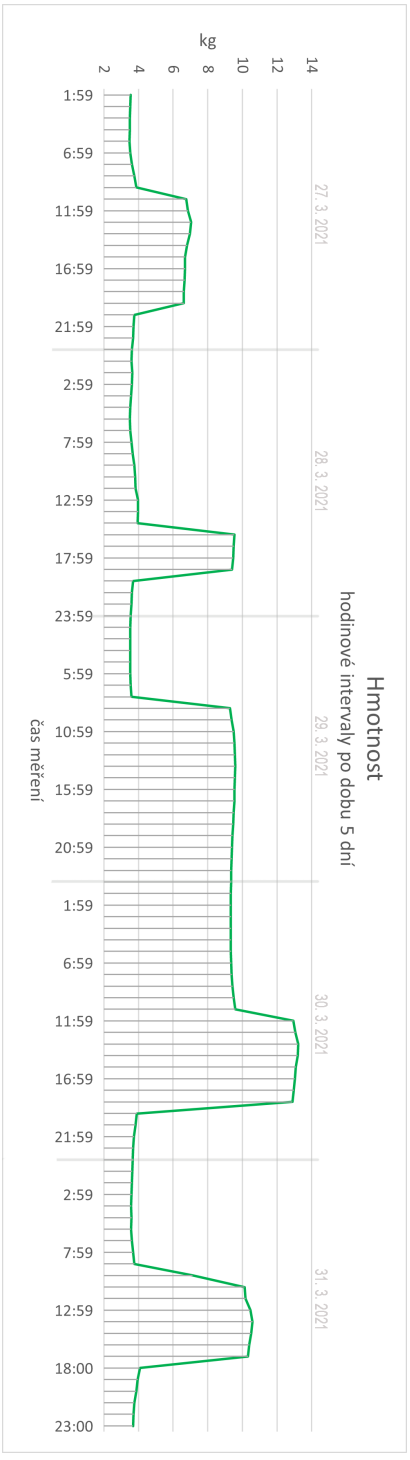
Vizualizace naměřených dat v grafech



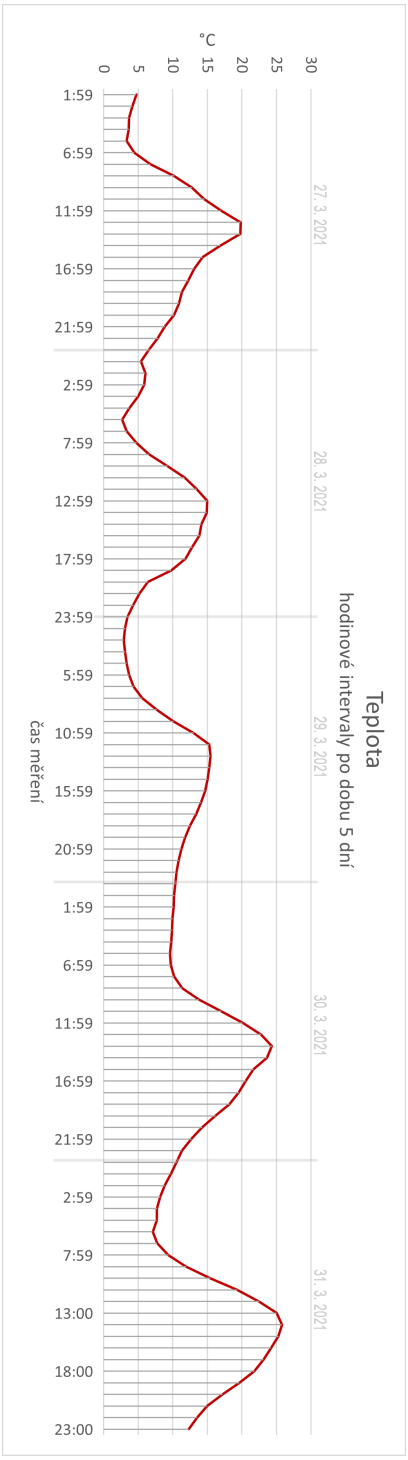
Obrázek C.1: Měřený tlak i s chybně zachycenými daty



Obrázek C.2: Vlhkost vzduchu i s chybně zachycenými daty

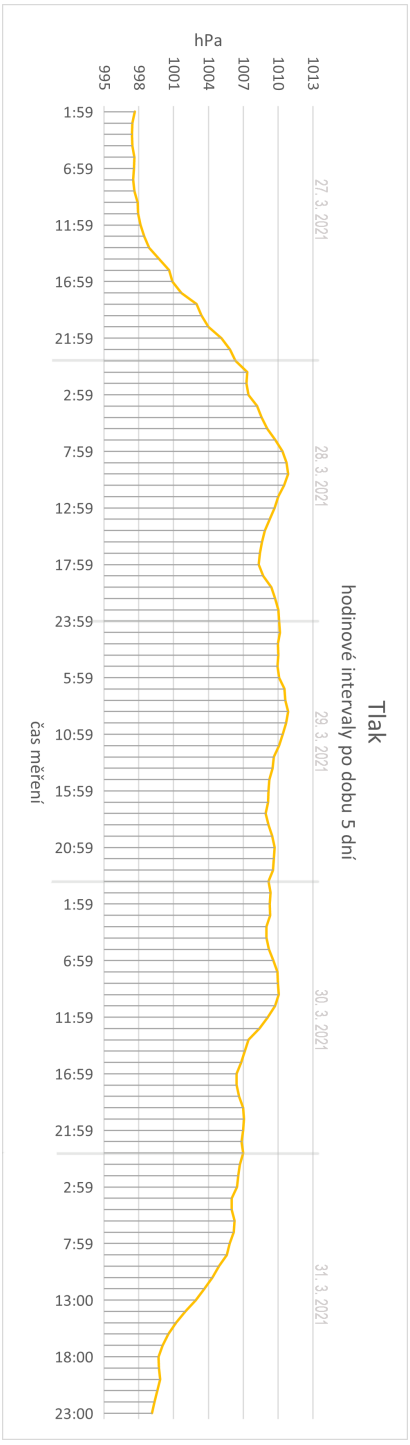


(a) Hmotnost na váze

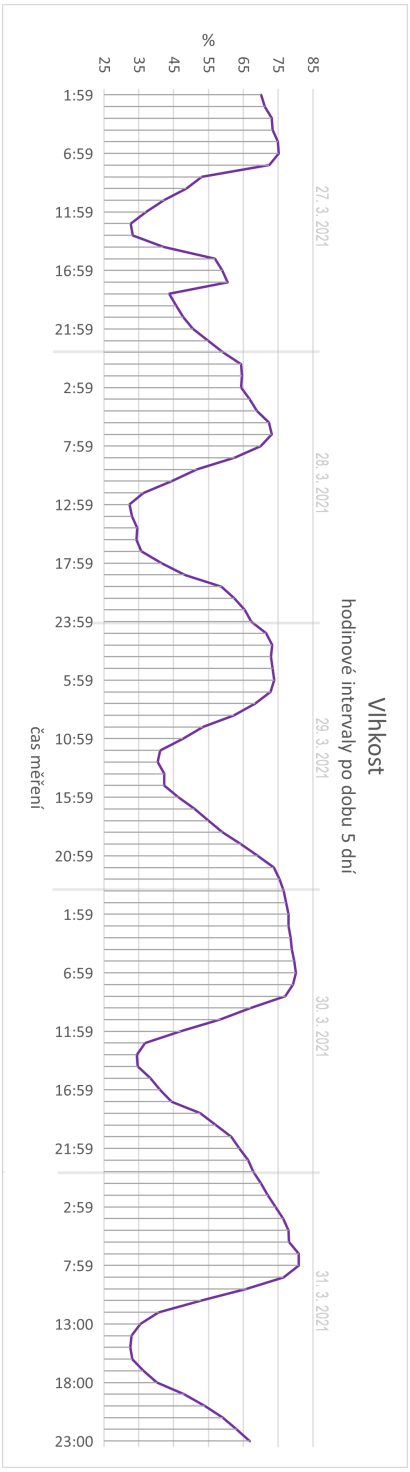


(b) Teplota vzduchu

Obrázek C.3: Pětidenní data - hmotnost na váze a teplota vzduchu

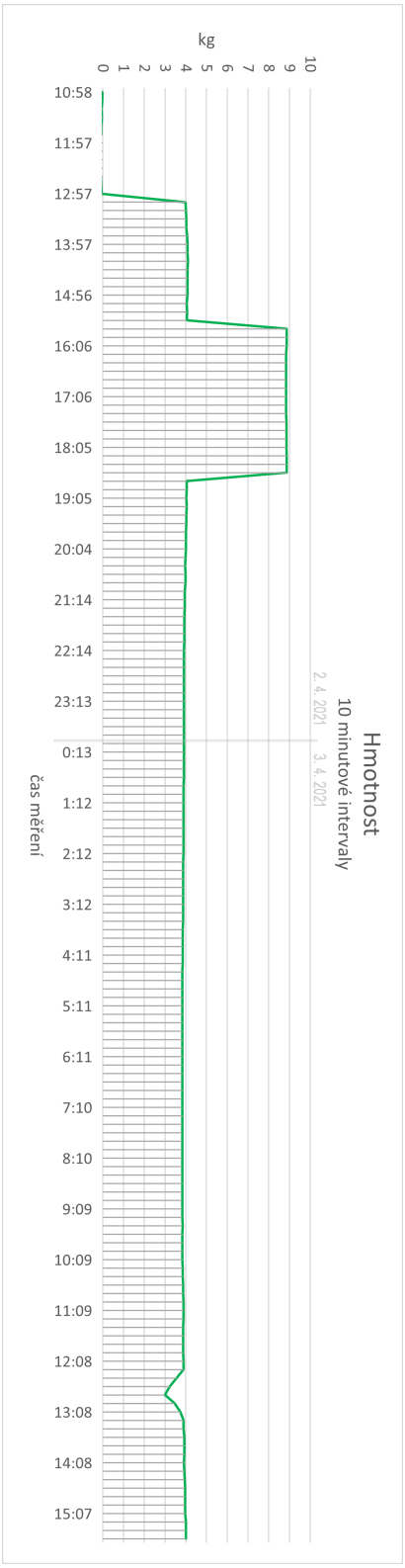


(a) Atmosférický (barometrický) tlak

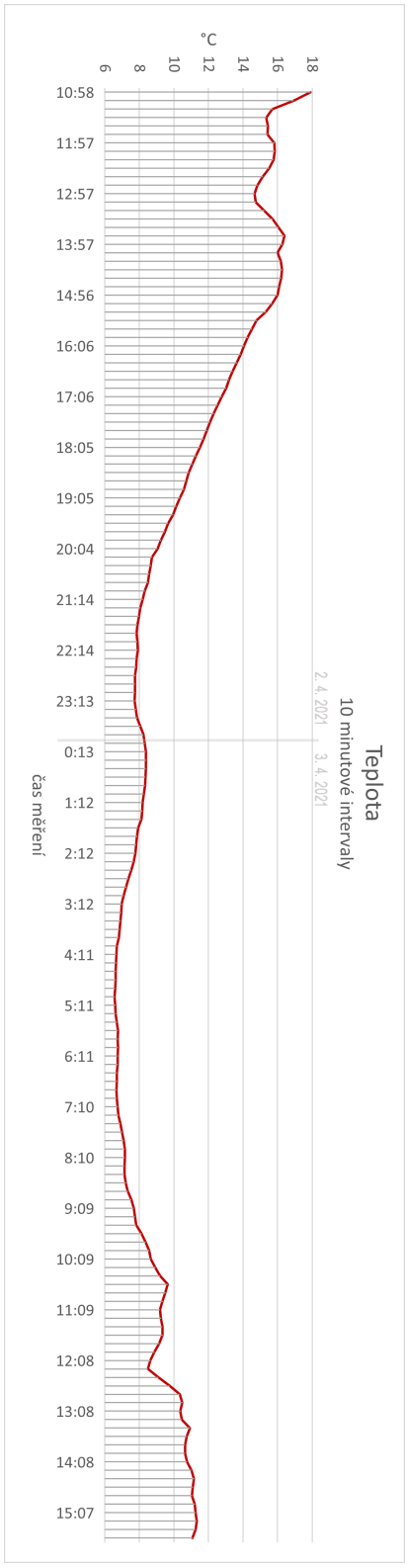


(b) Vlhkost vzduchu

Obrázek C.4: Pětidenní data - atmosférický tlak a vlhkost vzduchu

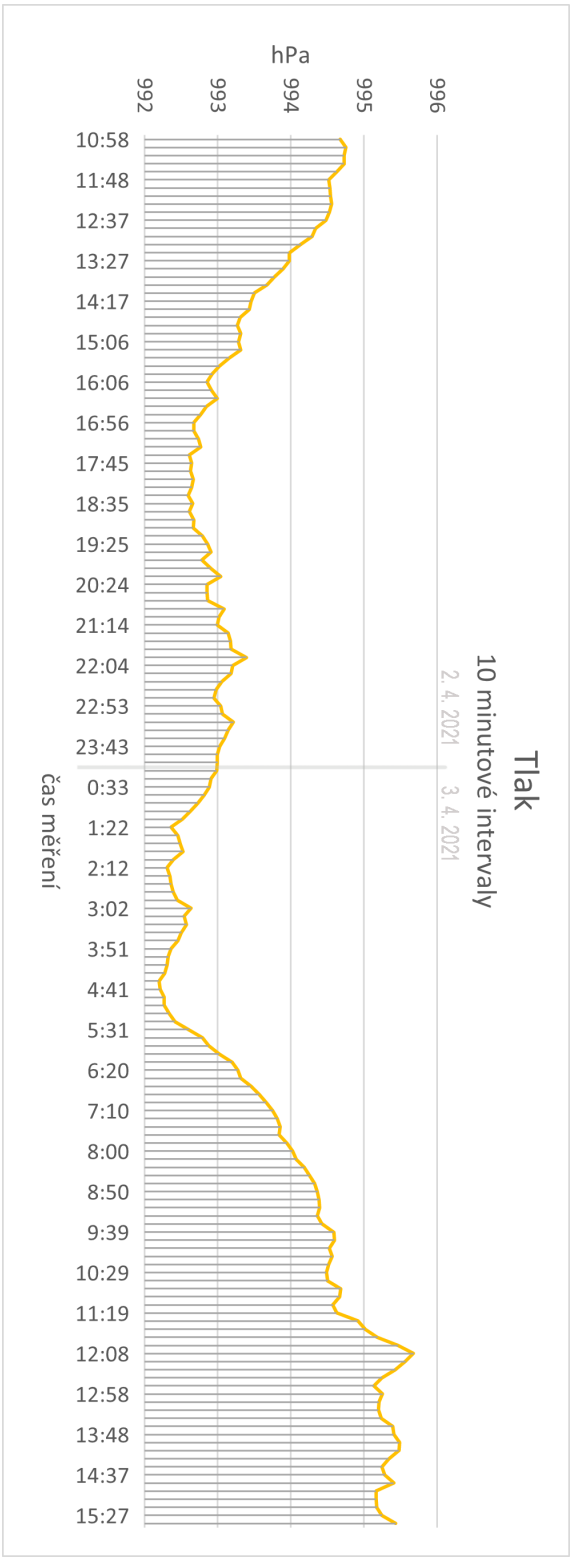


(a) Hmotnost na váze

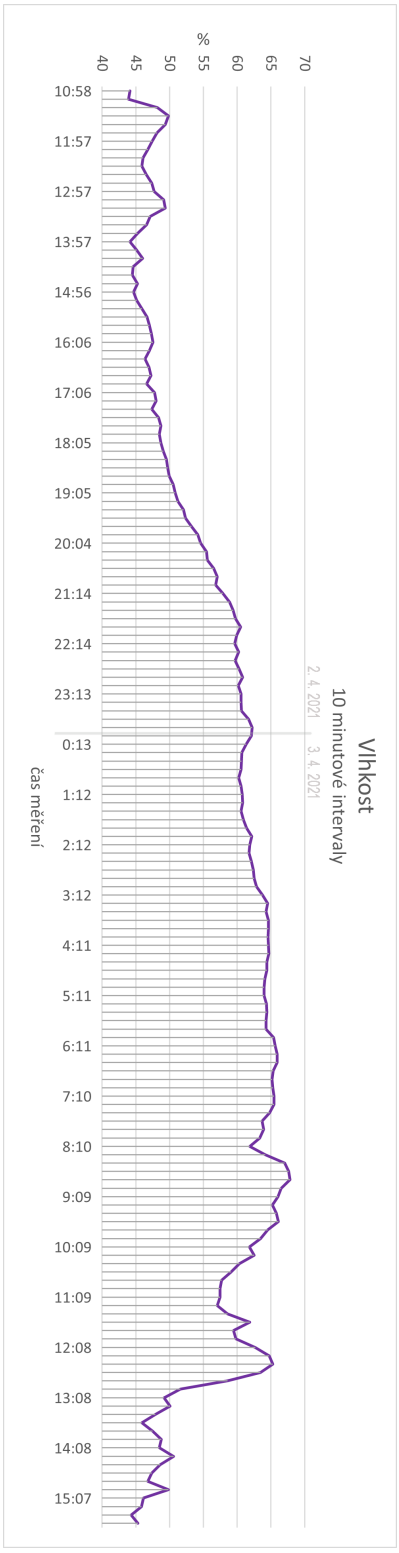


(b) Teplota vzduchu

Obrázek C.5: Desetiminutové intervaly - hmotnost na váze a teplota vzduchu



(a) Atmosférický (barometrický) tlak



(b) Vlhkost vzduchu

Obrázek C.6: Desetiminutové intervaly - atmosférický tlak a vlhkost vzduchu

Příloha D

Zdrojové kódy

Hlavičkový soubor pro práci s modulem BME280

```
#ifndef BP_BME280_HPP
#define BP_BME280_HPP

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include "bp_measurements.hpp"

#define SEALEVELPRESSURE_HPA (1013.25)

// BME280 - teplota, vlhkost, tlak
Adafruit_BME280 bme;

void BME280_Setup()
{
  // BME280
  if (!bme.begin(0x76)) {
    Serial.println("Error BME280 sensor!");
    // Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
}

Measurements getMeasurements(){
```

```
    Measurements m;
    m.temperature = bme.readTemperature();
    m.pressure = bme.readPressure() / 100.0f;
    m.altitude = bme.readAltitude(SEALEVELPRESSURE_HPA);
    m.humidity = bme.readHumidity();
    return m;
}

void printBME280()
{
    float temperature = bme.readTemperature();
    float pressure = bme.readPressure() / 100.0f;
    float altitude = bme.readAltitude(SEALEVELPRESSURE_HPA);
    float humidity = bme.readHumidity();

    Serial.println("Temperature = " + String(temperature) + " *C");
    Serial.println("Pressure = " + String(pressure) + " hPa");
    Serial.println("Approx. Altitude = " + String(altitude) + " m");
    Serial.println("Humidity = " + String(humidity) + " %");
    Serial.println();
}

#endif   // !BP_BME280_HPP
```

Listing D.1: bp_bme280.hpp

Hlavičkový soubor obsahující strukturu pro ukládání naměřených dat

```
#ifndef BP_MEASUREMENTS_HPP
#define BP_MEASUREMENTS_HPP

struct Measurements {
    float weight;        // Kg
    float temperature;   // *C
    float pressure;      // hPa
    float humidity;      // %
    float altitude;      // m
};

String createMessage(Measurements measurements)
{
    String msg = "";
    msg += " ";
    msg += "arduino_1";
    msg += " ";
    msg += measurements.weight;
    msg += " ";
    msg += measurements.temperature;
    msg += " ";
    msg += measurements.pressure;
    msg += " ";
    msg += measurements.humidity;

    return msg;
}

#endif   // !BP_MEASUREMENTS_HPP
```

Listing D.2: `bp_measurements.hpp`

Hlavičkový soubor pro práci s převodníkem HX711

```
#ifndef BP_SCALE_HX711_HPP
#define BP_SCALE_HX711_HPP

// RobTillaart/HX711 library
#include "HX711.h"

// HX711 circuit wiring
const int LOADCELL_DOUT = 4;
const int LOADCELL_SCK = 5;

const float SCALE_FACTOR = 2280.0f; // kalibracni faktor

HX711 scale;

void Scale_Setup()
{
   // Initialize library with data output pin, clock input pin and gain factor.
   // default "128" (Channel A) is used here.
   scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
   scale.set_scale(SCALE_FACTOR);
   scale.tare();                 // Vynulovani vahy
}

float getWeight()
{
    return scale.get_units() / 10;
}

void printScale()
{
     Serial.print("reading:\t");
     Serial.print(scale.get_units() / 10, 3);
     Serial.print("\t| average:\t");
     Serial.println(scale.get_units(10) / 10, 3);
}
```

```
#endif   // !BP_SCALE_HX711_HPP
```

Listing D.3: `bp_scale_hx711.hpp`

Hlavičkový soubor pro práci s modulem RFM95W

```
#ifndef BP_RFM95_HPP
#define BP_RFM95_HPP

// Lora RFM_95
#include <SPI.h>
#include <RH_RF95.h>
#include "bp_measurements.hpp"

#define RFM95_CS 10     // ss pin
#define RFM95_RST 9   // RST pin
#define RFM95_INT 2     // DIO0 pin

// frequency in Hz (434.0 for Asia, 868.0 for Europe, 915.0 for North America)
#define RF95_FREQ 868.0 // EU_FREQUENCY
#define MSG_LEN 41

// singleton instance of the radio driver
RH_RF95 rf95(RFM95_CS, RFM95_INT);

String createMessage(Measurements measurements);
void rfm95_send(Measurements mData);

void RFM95_Setup()
{
    pinMode(RFM95_RST, OUTPUT);
    digitalWrite(RFM95_RST, HIGH);

    Serial.println("Arduino LoRa RFM95W");

    // manual reset
    digitalWrite(RFM95_RST, LOW);
    delay(10);
```

```
    digitalWrite(RFM95_RST, HIGH);
    delay(10);

    while (!rf95.init()) {
        Serial.println("LoRa radio init failed");
        while (1);
    }
    Serial.println("LoRa radio init OK!");

    if (!rf95.setFrequency(RF95_FREQ)) {
        Serial.println("setFrequency failed");
        while (1);
    }

    // you can set transmitter powers from 5 to 23 dBm:
    rf95.setTxPower(18, false);
}

void rfm95_send(Measurements mData)
{
    // Vytvoreni zpravy
    String text = createMessage(mData);
    unsigned int text_len = text.length() + 1;
    char msg_buffer[text_len] = {0};
    text.toCharArray(msg_buffer, text_len);
    msg_buffer[text_len - 1] = 0;

    // Zaslani zpravy
    Serial.println("Sending...");
    delay(10);
    rf95.send((uint8_t *)msg_buffer, text_len);

    // Serial.println("Waiting for packet to complete...");
    delay(10);
    rf95.waitPacketSent();
}

void rfm95_waitForReply()
```

```
{
  // Cekani na odpoved
  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN] = {0};
  uint8_t len;

  Serial.println("Waiting for reply ...");
  delay(10);

  if (rf95.waitAvailableTimeout(WAIT_TIME_FOR_REPLY))
  {
     // Prijeti zpravy
     if (rf95.recv(buf, &len))
     {
        Serial.print("Got reply: ");
        Serial.println((char*)buf);
        Serial.print("RSSI: ");
        Serial.println(rf95.lastRssi(), DEC);
     }
     else
     {
        Serial.println("Receive failed");
     }
  }
  else
  {
     Serial.println("No reply, is there a listener around?");
  }
}

#endif   // !BP_SENDER_HPP
```

Listing D.4: `bp_rfm95.hpp`

Hlavní zdrojový kód běžící na Arduinu Nano

```
#include "bp_oled_ssd1306.hpp"
#include "bp_bme280.hpp"
#include "bp_rfm95.hpp"
#include "bp_scale_hx711.hpp"

void setup() {
  // while (!Serial);
  Serial.begin(9600);

  // Inicializace vahy HX711
  Scale_Setup();

  // Inicializace BME280 - teplota, vlhkost, tlak
  BME280_Setup();

  // Inicializace LoRa RFM95W
  RFM95_Setup();
}

void loop()
{
  float weight = getWeight();
  // printScale();

  Measurements mData = getMeasurements();
  mData.weight = weight;
  scale.power_down();   // put the ADC in sleep mode

  rfm95_send(mData);
  Serial.println("Sent");
  // rfm95_waitForReply();

   // Prodleva pred znovuopakovanim cyklu
   // 60 * (60 s * 1000 milisekund (1 sec)) = 1 min.) = 1 hodina v datovem typu
       unsigned long
   delay(60UL * 60UL * 1000UL - 5020UL); // pouzito pro hodinove intervaly
```

```
  // delay(10UL * 60UL * 1000UL - 5020UL); // pouzito pro desetiminutove
     intervaly
  scale.power_up();
}
```

Listing D.5: bp_sender_client.ino

Hlavní zdrojový kód běžící na WeMos D1 R2 zařízení

```
#include <LoRa.h>
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

const int csPin = 15;       // LoRa radio chip select
const int resetPin = 16;    // LoRa radio reset
const int irqPin = 5;       // 5 - DIO0, must be a hardware interrupt pin

const long frequency = 868E6; // EU LoRa Frequency
const int sync_word_length = 4;

/************************ WiFi Access Point *********************************/
#define WLAN_SSID     "SSID"
#define WLAN_PASS     "PASSWD"

/************************ Adafruit.io Setup *********************************/
#define AIO_SERVER    "io.adafruit.com"
#define AIO_SERVERPORT 1883                   // or use 8883 for SSL
// Adafruit IO username https://accounts.adafruit.com
#define AIO_USERNAME  "Username"
#define AIO_KEY       "aio_eiCm40uquRLGw3MXKdrZjj7MU481"

// ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME,
    AIO_KEY);

/****************************** Feeds ***************************************/
```

```
// Setup a feeds for publishing.
// MQTT paths for AIO follow the form: <username>/feeds/<feedname>
Adafruit_MQTT_Publish hmotnost = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds
    /arduino-1.hmotnost");
Adafruit_MQTT_Publish teplota = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/
    arduino-1.teplota");
Adafruit_MQTT_Publish vlhkost = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/
    arduino-1.vlhkost");
Adafruit_MQTT_Publish tlak = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/
    arduino-1.tlak");

void onReceive(int packetSize);
void sendMessage(String outgoing);
void MQTT_connect();

void setup()
{
  Serial.begin(115200);
  delay(1000);

  /* while (!Serial){
      Serial.println("Serial failed!");
      delay(1000);
  }; */

  Serial.println("LoRa Receiver init");
  LoRa.setPins(csPin, resetPin, irqPin);

  if (!LoRa.begin(frequency)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  Serial.println("LoRa init succeeded.");

  // Pripojeni k WiFi pristupovemu bodu (AP)
  Serial.println(); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);
```

```
  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop()
{
  // Zavolani onReceive s rozparsovanym paketem
  onReceive(LoRa.parsePacket());
}

void onReceive(int packetSize)
{
   if (packetSize == 0) return;   // Test, pokud neni zadny paket

   MQTT_connect();

   Serial.print("Received packet '");
   String receivedText;

   // Cteni paketu
   while (LoRa.available()) {
      receivedText += (char)LoRa.read();
   }
   Serial.print(receivedText);

   // Vypsai RSSI paketu
   Serial.print("' with RSSI ");
   Serial.println(LoRa.packetRssi());
```

```
Serial.println("sync_word message in hexa:");
Serial.print(receivedText[0], HEX);
Serial.print(receivedText[1], HEX);
Serial.print(receivedText[2], HEX);
Serial.print(receivedText[3], HEX);

char v_device[20] = {0};
float v_hmotnost, v_teplota, v_tlak, v_vlhkost;

// Parsovani prijate zpravy a ulozeni do promennych
int n = sscanf(receivedText.c_str() + sync_word_length, "%s %f %f %f %f",
    v_device, &v_hmotnost, &v_teplota, &v_tlak, &v_vlhkost);
if (n != 5 || String(v_device) != "arduino_1"){
    return;
}

// rfm95_sendReply();

Serial.print(F("\nGot message from device: "));
Serial.println(v_device);

// Publishovani na MQTT broker server
Serial.print(F("\nZasilam hmotnost "));
Serial.println(v_hmotnost);

if (!hmotnost.publish(v_hmotnost)) {
   Serial.println(F("Failed"));
} else {
   Serial.println(F("OK!"));
}

Serial.print(F("\nZasilam teplotu "));
Serial.println(v_teplota);
if (!teplota.publish(v_teplota)) {
   Serial.println(F("Failed"));
} else {
   Serial.println(F("OK!"));
}
```

```
    Serial.print(F("\nZasilam tlak "));
    Serial.println(v_tlak);
    if (!tlak.publish(v_tlak)) {
       Serial.println(F("Failed"));
    } else {
       Serial.println(F("OK!"));
    }

    Serial.print(F("\nZasilam vlhkost "));
    Serial.println(v_vlhkost);
    if (!vlhkost.publish(v_vlhkost)) {
       Serial.println(F("Failed"));
    } else {
       Serial.println(F("OK!"));
    }
}

// Funkce na pripojeni anebo znovu pripojeni
void MQTT_connect() {
  int8_t ret;

  // stop if already connected.
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000); // wait 5 seconds
       retries--;
       if (retries == 0) {
         // basically die and wait for WDT to reset me
```

```
        while (1);
      }
  }
  Serial.println("MQTT Connected!");
}

void rfm95_sendReply()
{
    // Zaslani odpovedi
    Serial.println("Sending reply message");

    byte syncWord[2] = {0xFF, 0xFF};
    String syncWordTex = (char*)syncWord;
    String text = syncWordTex + " Got message";

    Serial.println("Sending reply: " + text);
    delay(10);

    LoRa.beginPacket();
    LoRa.print(text);
    LoRa.endPacket();
    //Serial.println("Reply message sent");
}
```

Listing D.6: `wemos_server_side.ino`